VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

3D modelování objektů v systému FOTOM^{NG}

3D modeling of buildings using FOTOM^{NG}

2015 Dalibor Lis

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Dalibor Lis**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: 3D modelování objektů v systému FOTOM-NG
3D modeling of buildings using FOTOM-NG

Zásady pro vypracování:

Cílem diplomové práce je návrh a realizace modulu pro 3D modelování objektů s použitím voxelové grafiky ze série snímků. Dále návrh a realizace rekonstrukce objektu pomocí Radonové transformace. Součástí práce bude také vyhodnocování základních vlastností rekonstruovaných objemových dat (objem, plocha průřezu, atd.).
1. Seznamte se s problematikou počítačového zpracování fotografií a metod 3D modelování objektů požitím Voxelové grafiky a Radonové transformaci ze série snímků.
2. Seznamte se s prostředním NetBeans a programovacím jazykem JAVA.
3. Seznamte s fotogrammetrickým systémem FOTOM-NG.
4. Seznamte se s metodou rekonstrukce objektu pomocí Radonovi transformace.
5. Navrhněte nový modul na 3D modelování objektů s použitím Voxelové grafiky ze série snímků a nový modul na rekonstrukci objektu s použitím Radonovy transformace ze série snímků v rámci systému FOTOM-NG.
6. Implementujte navržené moduly v systému FOTOM-NG a proveďte zhodnocení dosažených výsledků.
7. Vypracujte uživatelskou a programátorskou dokumentaci.

Seznam doporučené odborné literatury:

[1] Heiko Böck: NetBeans Podrobný Průvodce Programátora, 2010, Computer Press, Czech Republic, ISBN: 978-80-251-3116-9, 320 Pages
[2] Heiko Böck: The Definitive Guide to the NetBeans Platform 6.5, May 2009, Apress, USA ISBN: 978-1-4302-2417-4, 450 Pages
[3] Ličev, Lačezar: Analýza, modelování, rozpoznávání a vizualizace procesu měření objektů na snímcích, 128 str., Knihy vydané prostřednictvím www.vydejteknihu.cz, Computer Press, a.s., ISBN 978-80-2513-296-8, EAN 9788025132968
[4] Vlček, Vítězslav Vít a Segeth, Karel: Matematika dokonale ukrytá v počítačové tomografii. Pokroky matematiky, fyziky a astronomie, Vol. 53 (2008), No. 3, 199—210, dostupné na http://dml.cz/dmlcz/141859
[5] HLAVÁČ, Václav, SEDLÁČEK, Miloš: Zpracování signálu a obrazu, Pracovní verze skripta v tisku pro studenty, 1999, 116 str.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Lačezar Ličev, CSc.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. dubna 2015 .....

Na tomto místě bych chtěl poděkovat vedoucímu diplomové práce panu doc. Ing. Lačezaru Ličevovi CSc. za rady a připomínky při tvorbě této práce.

Abstrakt

Cílem práce je návrh a implementace dvou nových modulů, které rozšíří funkcionalitu systému FOTOM^{NG}. První modul slouží k rekonstrukci objektu zájmu v řezech ze série snímků pomocí algoritmu inverzní Radonovy transformace. Druhý modul je určen k 3D modelování objemových dat pomocí vybraných vizualizačních algoritmů. Objemová data jsou reprezentována v podobě série snímků řezů objektu a mohou být vytvořena pomocí algoritmu inverzní Radonovy transformace, který implementuje první modul. Oba moduly byly otestovány na reálných datech. Součástí implementovaných modulů je programátorská a uživatelská dokumentace. Vytvořené moduly jsou implementovány v programovacím jazyku Java nad platformou NetBeans. Součástí práce je zhodnocení dosažených výsledků.

Klíčová slova: objemová data, Radonova transformace, 3D modelování, NetBeans, Java, FOTOM^{NG}, modul

Abstract

The aim of this work is design and implementation of two new modules, which extends functionality of system FOTOM^{NG}. The first module is intended to reconstruction object of interest in cuts from series of images by using algorithm of inverse Radon transform. The second module is intended to 3D modeling of volumetric data by using selected visualisation algorithms. Volumetric data are represented as series of images containing cuts of object and can be created by algorithm of inverse Radon transform, which is implemented by first module. The both modules were tested on real data. As a part of implemented modules is programming documentation and user documentation. Created modules are implemented in Java programming language over NetBeans platform. Additional part of work is evaluation of reached results.

Keywords: volumetric data, Radon transform, 3D modeling, NetBeans, Java, FOTOM^{NG}, module

Seznam použitých zkratek a symbolů

2D	–	Two Dimensional
3D	–	Three Dimensional
BMP	–	Bitmap
CT	–	Computed Tomography
GIF	–	Graphics Interchange Format
GPU	–	Graphics Processing Unit
JDK	–	Java Development Kit
JNI	–	Java Native Interface
JOGL	–	Java OpenGL
JPEG	–	Joint Photographic Experts Group
MVC	–	Model View Controller
OpenGL	–	Open Graphics Library
PNG	–	Portable Network Graphics

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Trojrozměrná počítačová grafika v současnosti představuje velmi častý způsob, jak lze vizualizovat výstup různých vědeckých simulací, technických měření či výstup jiných programů v 3D podobě. Výpočetní výkon počítačů se neustále zvyšuje a umožňuje nám tak provádět počítačovou 3D vizualizaci ve stále detailnější podobě a také nám umožňuje vizualizovat data, která svými vlastnostmi kladou v průběhu vizualizace vysoké nároky na výpočetní výkon a dostupnou dynamickou paměť počítače. Pro modelování 3D objektů existuje více vizualizačních technik, které se od sebe liší na základě pojetí reprezentace těles. Nejčastěji bývají používány metody založené na hraniční reprezentaci těles, kde jsou tělesa popsána hranami a vrcholy. Ve specializovaných nástrojích pro projektování je naopak častěji používána reprezentace pomocí konstruktivní geometrie pevných těles. Existují však i typy dat, které svými vlastnostmi nejsou snadno geometricky definovatelné a představují tak problém pro vizualizaci pomocí uvedených technik. Jedním z typů takových dat jsou data objemová, která jsou tvořena množinou vzorků bodů. Objemová data mohou být vytvářena, jako výstup různých simulací, častěji se však s nimi můžeme setkat, jako s výstupem počítačové tomografie v podobě jednotlivých CT snímků. Pro vizualizaci objemových dat je nutné zvolit sofistikovanější způsob, jak data popsat a zobrazit. Pro tyto účely existují vybrané algoritmy, které jsou založeny na fyzikálním principu průchodu paprsku světla skrz objemová data a následném vyčíslení objemového integrálu, který průchod popisuje. Na základě uvedeného principu jsme schopni objemová data popsat a dále vizualizovat.

Jak bylo uvedeno výše, častým zdrojem výstupu objemových dat jsou snímky počítačové tomografie. Způsob pořízení snímků počítačovým tomografem je založen na principu Radonovy transformace. Rentgenka tomografu obíhá okolo pacienta po určené dráze a pod různými úhly pořizuje rentgenové snímky. K jednotlivým snímkům tak známe úhly, pod kterými byly pořízeny. Získané snímky spolu s úhly slouží, jako vstup pro Radonovu transformaci, která akumuluje součty intenzit jasů podél přímek, které byly pod danými úhly vrhány skrz snímky. Uvedený princip není nijak omezen pouze na počítačovou tomografii a zdravotnictví a lze jej tak aplikovat i pro snímání jiných objektů. Radonova transformace má i svou inverzní variantu, která může být využita pro zpětnou rekonstrukci snímků v podobě, jakou nám poskytuje počítačový tomograf, avšak pro libovolný rekonstruovaný objekt. Tímto způsobem lze vytvářet objemová data pro následnou 3D vizualizaci pomocí vhodných vizualizačních algoritmů.

Cílem této práce je popis a praktická implementace vybraných vizualizačních algoritmů objemových dat v 3D podobě a také algoritmu inverzní Radonovy transformace, který bude využit k efektivnímu způsobu, jak získat objemová data vybraného objektu zájmu. Součástí praktické části je také implementace vybraných algoritmů zpracování obrazu, které umožní přesnější detekci objektů zájmu. Praktická část práce bude přidána v podobě modulů do fotogrametrického systému FOTOM^{NG}, který tak bude rozšířen o novou funkcionalitu. Nově přidaná fukcionalita bude otestována na reálných datech. Textová část práce je rozdělena do několika kapitol. V prvních čtyřech kapitolách jsou popsány teoretické principy a východiska realizace zadaného problému, která jsou ná-

sledně použita pro praktickou realizaci modulu vizualizace objemových dat a modulu inverzní Radonovy transformace. Následující kapitoly popisují samotnou praktickou realizaci modulů. Kapitoly obsahují popis technických aspektů realizace modulů, spolu s úryvky zdrojových kódů a obrázky, znázorňující grafický výstup modulů. V předposlední části je provedeno zhodnocení dosažených výsledků s ohledem na stanovené body zadání práce. Poslední kapitola obsahuje závěr, ve kterém je práce shrnuta, jako celek.

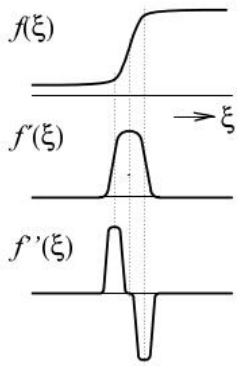
2 Segmentace obrazu

2.1 Úvod

Segmentací obrazu rozumíme operaci, při které se snažíme z obrazové funkce oddělit objekty zájmu, také označované jako objekty popředí, od objektů pozadí. Segmentace obrazu není tvořena pouze jednou samostatnou metodou, ale soustavou různých druhů metod, které slouží ke stejnému účelu a to k extrakci objektů zájmu pomocí odlišných technik. V zásadě lze rozdělit segmentační metody do dvou základních skupin. První skupinu tvoří metody založené na detekci objektů pomocí detekce jejich hran, druhou skupinu pak tvoří metody založené na detekci celých oblastí. Pro obě uvedené skupiny jsou věnovány následující podkapitoly.

2.2 Detekce hran

Při detekci hran předpokládáme, že každý objekt je tvořen souvislou oblastí, která je ohraničena hranou. Hrany pak tvoří hranice mezi jednotlivými objekty. Pro detekci hran existuje více postupů. Většina postupů detekce hran je založena na skutečnosti, že v místě hrany je absolutní hodnota první derivace funkce průběhu jasu rovna vysokým hodnotám. [4] Souhrnně tyto metody nazýváme gradientními metodami. Ukázka typického průběhu jasu ve směru napříč hranou včetně první a druhé derivace můžeme vidět na obrázku číslo 1. Hodnotě derivace též říkáme velikost hrany.



Obrázek 1: Průběh hrany včetně první a druhé derivace [4]

Pro určení velikosti hrany používáme hranové operátory, které jsou většinou aplikovány na každý bod obrazu. Jako nejjednodušší hranový operátor můžeme použít derivaci jasové funkce ve směru osy x a derivaci ve směru osy y. Pokud bychom však pracovali pouze s takto jednoduše definovanými hranovými operátory, dokázali bychom detekovat jen hrany rovnoběžné s osou x nebo osou y. Pro zobecnění na hrany libovolného směru je nutné vyšetřovat průběh jasu ve směru kolmém na směr potenciální hrany. Směr potenciální hrany není ve většině případů znám, ale je znám fakt, že směr hrany je vždy kolmý ke směru gradientu. [4] To nám umožňuje určit směr hrany na základě

hodnot jasové funkce. Vektor popisující směr kolmý ke směru potenciální hrany odpovídá vztahu $n = (cos\theta, sin\theta)$ a ζ odpovídá souřadnici měřené v tomto směru. Derivaci ve směru kolmo k hraně popisuje vztah číslo 1.

$$\frac{\partial f}{\partial \zeta} = grad(f) \cdot n = \frac{\partial f}{\partial x} cos\theta + \frac{\partial f}{\partial y} sin\theta \tag{1}$$

Velikost hrany odpovídá ze vztahu 1 hodnotě $|\partial f / \partial \zeta|$. Pro stanovení, zda ve vyšetřovaném bodě je či není hrana je rozhodující směr, v němž je změna jasu největší. V směru s nejvyšší změnou jasu se pak nachází gradient obrazové funkce. Směr hrany je kolmý na směr gradientu, velikost hrany odpovídá velikosti gradientu. Z uvedených skutečností platí následující vztahy:

$$e(x, y) = \sqrt{f_x^2(x, y) + f_y^2(x, y)} \tag{2}$$

$$\varphi(x, y) = arctan\left[\frac{f_x(x, y)}{f_y(x, y)}\right] \tag{3}$$

$$\psi(x, y) = \varphi(x, y) + \frac{\pi}{2} \tag{4}$$

Kde:

$e(x, y)$ je velikost hrany
$\varphi(x, y)$ směr gradientu
$\psi(x, y)$ značí směr hrany v bodě (x, y)
$f_x(x, y)$ je zkrácený zápis pro $\frac{\partial f}{\partial x}$
$f_y(x, y)$ je zkrácený zápis pro $\frac{\partial f}{\partial y}$

Uvedené vztahy jsou platné pouze pro spojitou obrazovou funkci. V praxi však pracujeme s diskrétními hodnotami. Pro použití uvedených vztahů v diskrétní variantě je nutné nahradit derivaci pomocí diferencí.[4] Úprava pro diskrétní případ je uvedena v následujících vztazích:

$$f_x(x, y) = f(x + 1, y) - f(x, y) \tag{5}$$

$$f_y(x, y) = f(x, y + 1) - f(x, y) \tag{6}$$

Po úpravách podle vztahů 5 a 6 lze uvedené vztahy aplikovat na obrazovou funkci v diskrétní variantě. Na základě získaní hodnoty $e(x, y)$ je stanoveno, zda daný bod spadá do oblasti hrany okolo daného objektu. Častým postupem je stanovení pevně dané prahové hodnoty, se kterou je hodnota $e(x, y)$ porovnána a v případě, kdy je $e(x, y)$ větší než stanovená prahová hodnota, považujeme daný bod za bod hrany. Problematice prahování je věnovaná podkapitola 2.3 na straně 9.

Při detekování hran, jakožto hranic mezi objekty, můžeme narazit na problémy neúplné hranice či problém chybně detekovaného bodu, jako bodu hranice. V případě problému neúplné hranice může být obraz upraven pomocí operací matematické morfologie, která řeší problém neúplné hrany pomocí operace dilatace, ale také problém příliš

široké hrany pomocí operace eroze. Širokou hranou je myšlena hrana, která je složena s více bodů (pixelů). Intuitivně chápeme hranu, jako tenkou linii tvořenou pouze jedním bodem na šířku, nicméně při počítačovém detekování hrany nejsme schopni zajistit, aby nebyly detekované hrany tvořeny více body na šířku. Krom operací matematické morfologie rovněž existují složitější hranové detektory, které se snaží zmíněné problémy odstranit v průběhu detekce. Důležité je také zmínit problém s obrazy obsahujícími značné množství šumu, kde se přítomnost šumu negativně projevuje při detekci hran. Šum představuje vysokofrekvenční hodnoty ve frekvenčním spektru. Ve stejné oblasti vysokých hodnot frekvenčního spektra se rovněž nacházejí hodnoty bodů hran. Odstranění šumu není úplně jednoduchou operací, protože při potlačení šumu zároveň dochází ke zhoršení detekce hran. Je tedy nutné stanovit vhodnou úroveň potlačení šumu tak, aby nedošlo k omezení detekce hran. Pro odstranění šumu se často používají obrazové filtry, které jsou aplikovány pomocí operace konvoluce. Často používaným filtrem k potlačení šumu je Gaussův filtr, který provádí rozmazání obrazu a tím potlačuje šum.

Uvedené informace o problematice detekce hran byly čerpány ze zdroje [4].

2.3 Prahování

Prahování patří do segmentačních metod, které pracují na principu detekce celých oblastí. Předpokladem pro úspěšnou detekci oblasti je homogenita detekované oblasti. Homogenitou rozumíme konstantní rozložení vybrané veličiny, která rovnoměrně pokrývá oblast. Mezi takové veličiny, které často bývají konstantně rozloženy v detekované oblasti, patří hodnota jasu, barvy či výplň texturou. Pokud bychom tuto metodu segmentace přímo porovnávali, jako rovnocennou metodu k metodám detekce hran, nebylo by toto porovnání příliš správné. Jak bylo v předchozím textu uvedeno, metody detekce hran jsou velmi citlivé na přítomnost šumu v obrazu. Pro velmi nekvalitní obrazy obsahující velké množství šumu, je lepším řešením detekce oblastí pomocí metody prahování. V těchto obrazech je vyšší pravděpodobnost, že narazíme na homogenitu oblastí, která umožní jejich detekci. Naopak u obrazů s nízkou úrovní šumu mohou metody založené na detekci hran podávat lepší výsledky.

Metoda prahování patří k nejjednodušším metodám, které lze použít pro detekci celých oblastí a současně která za vhodných podmínek poskytuje dobré výsledky. Přesnějším označením je název binární prahování obrazu. Název binární je zde z důvodu skutečnosti, že vstupní obraz, ve kterém dochází k detekci oblastí, je po aplikování metody prahování převeden na binární výstupní obraz. Binární obraz obsahuje hodnoty logická 1 v bodech, kde se nachází detekovaná oblast a hodnotu logická 0 v bodech ostatních. Detekce oblasti probíhá za předpokladu konzistentních hodnot jasů (či jiné veličiny) jednotlivých bodů oblasti. Nejčastěji porovnávanou hodnotou, podle které dochází k oddělení oblasti od pozadí, je jasová hodnota. Za bod oblasti považujeme bod, jehož jasová hodnota spadá do určeného intervalu $< a, b >$. V praktických aplikacích je často postupováno tak, že je stanovena jedna prahová hodnota t, podle které dochází k rozhodnutí, kam daný bod zařadit. Princip je založen na porovnání aktuálně zpracovávané hodnoty bodu s touto prahovou hodnotou t a pokud je úroveň jasu zpracovávaného bodu vyšší

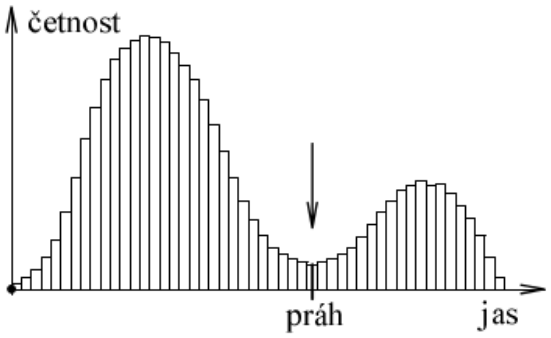
nebo rovna hodnotě prahu, je bod považován za bod oblasti, pokud je naopak nižší, je bod považován za bod pozadí. Celý princip je popsán následujícím vztahem:

$$g(x,y) = \begin{cases} 1, f(x,y) \geq t \\ 0, f(x,y) < t \end{cases} \tag{7}$$

Kde:

$g(x,y)$ je hodnota bodu binárního obrazu v souřadicích x a y
$f(x,y)$ je hodnota bodu původního obrazu v souřadicích x a y
t je prahová hodnota

Obecným problémem je stanovení vhodné hodnoty t, při které dochází ke správné detekci oblastí a pozadí. Hodnotu lze stanovit experimentálně zkoušením nebo na základě uživatelských zkušeností s daným typem obrazů. Pokud nejsme schopni určit vhodnou hodnotu prahu t, lze využít obecného doporučení, kdy v případě obrazů s bimodálním histogramem rozdělení jasů volíme minimální hodnotu ležící mezi oběma vrcholy. Ukázku takového bimodálního histogramu lze vidět na obrázku číslo 2.



Obrázek 2: Ukázka bimodálního histogramu [4]

Bimodální rozdělení vzniká za předpokladu, kdy v obrazu existují dva druhy pixelů (bodů), které mají četné zastoupení. Díky četnosti zastoupení vznikají dva vrcholy, mezi kterými poté leží minimum. Pokud se zaměříme na konkrétní popis obrazu, pak body oblasti zabírají vybranou část obrazu a body pozadí zabírají zbylou část obrazu. Často se můžeme setkat s obrazy, které bimodální rozdělení jasů nemají. Mezi takové obrazy patří obrazy s nerovnoměrným rozdělením jasu. V takových případech je nutné přejít k metodě prahování s proměnlivou hodnotou prahu. Prahování poté probíhá s dynamicky se měnící hodnotou prahu podle části obrazu, ve které prahování aktuálně probíhá. Při praktické realizaci prahování s proměnlivou hodnotou prahu rozdělíme obraz na několik částí a poté v každé části provádíme lokální prahování. Hodnotu prahu t v aktuálně zpracovávané oblasti můžeme stanovit na základě bimodálního histogramu rozložení jasových hodnot pro daný blok anebo v případě absence bimodálního histogramu zvolíme hodnotu prahu t, jako průměr prahových hodnot sousedních bloků.[4]

2.3.1 Ostatní metody detekce celých oblastí

Kromě metody prahování existují i jiné metody, které slouží k detekci celých oblastí. Mezi tyto metody patří:

- Metoda spojování oblastí
- Metoda dělení oblastí

Uvedené metody nejsou vhodné pro zamýšlenou aplikaci, a proto zde nejsou dále podrobněji rozepsány.

2.4 Matematická morfologie

Matematickou morfologii řadíme do zvláštní skupiny metod, které jsou určeny pouze pro práci s binárními obrazy. Operace, které tyto metody provádějí, jsou určeny pouze pro dvouhodnotové (binární) obrazy a nelze je aplikovat na jiné typy obrazů. Binární obraz bývá často výstupem segmentačních metod, nejčastěji je produkován segmentační metodou binárního prahování. Základní myšlenkou matematické morfologie je aplikování předem známé binární masky na vstupní binární obraz a následné provedení modifikace obrazu podle zvolené operace. Matematickou morfologii tvoří dvě základní operace, které mohou být aplikovány v různém pořadí, případně mohou být kombinovány.

Základními operacemi matematické morfologie jsou:

- Dilatace - značíme **D**
- Eroze - značíme **E**

Uvedené operace jsou definovány následovně:

$$\begin{aligned} E = B \otimes S = \{x, y | S_{x,y} \subseteq B\} \\ D = B \oplus S = \{x, y | S_{x,y} \cap B \neq \emptyset\} \end{aligned} \tag{8}$$

Kde:
B označuje vstupní binární obraz
S označuje binární masku
$S_{x,y}$ označuje bod masky o souřadnicích $[x, y]$

Z matematické formulace vidíme, že v případě operace eroze je bod výstupního obrazu označen za binární jedna tehdy, když jsou všechny body binární masky podmnožinou vstupního binárního obrazu. Obecněji formulováno, v případě, kdy jsou všechny body masky obsahující hodnotu binární jedna na pozicích vstupního obrazu, které rovněž obsahují hodnoty binární jedna, je výsledkem operace binární jedna ve výstupním obrazu. V ostatních případech je výstupem binární 0. Pro operaci dilatace platí, že bod výstupního obrazu je označen jako binární jedna, pokud je průnik hodnot mezi binární maskou a vstupním obrazem neprázdný. Z této definice vyplývá, že stačí pouze jeden bod, který

nese v binární masce hodnotu jedna a současně na stejné pozici ve vstupním obraze také hodnotu jedna, pak je výsledkem operace jedna. Na základě definice operace eroze vidíme, že tato operace slouží k ztenčování hrubých objektů, případně k eliminaci tenkých objektů. Operace dilatace naopak tenké objekty rozšiřuje a slouží k propojení přerušených hran objektů.[4]

2.4.1 Varianty použití

Uvedené operace matematické morfologie lze různě kombinovat. Lze použít několik operací dilatace za sebou a poté provést stejný počet operací eroze. Pořadí, kombinace a počet aplikování není nijak omezen, záleží na daném typu řešeného problému. Mezi časté operace matematické morfologie, které jsou složeny z operací eroze a dilatace patří operace uzávěru a operace otevření. Operace otevření je definována, jako provedení operace eroze následované operací dilatace. Uzavření je definováno opačným postupem, kdy je na vstupní binární obraz aplikována jako první operace dilatace a následně operace eroze. Operace otevření je vhodná pro odstranění malých objektů v obraze a rozdělení velkých objektů v místech, kde mají přerušenou hranu. Operace uzavření naopak slouží k propojení objektů v místech přerušených hran a vyplnění děr v objektech.

3 Radonova transformace

3.1 Úvod

Radonovou transformací rozumíme matematickou transformaci, která je postavena na integrální transformaci. Hlavní princip transformace spočívá v integraci funkce přes přímky. Transformace byla formulována rakouským matematikem Johannem Karl Gustav Radonem v roce 1917, který rovněž formuloval její inverzní variantu. V současnosti je Radonova transformace (a její inverzní varianta) využívána v počítačové tomografii (CT) k rekonstrukci snímaných objektů.[1]

3.2 Matematická formulace

Z předchozího textu vyplývá, že se jedná o matematickou transformaci postavenou na integraci funkce přes přímky. Pro vyjádření transformace ve spojitém tvaru existuje více ekvivalentních zápisů. Uveďme zde variantu vhodnou pro obrazovou funkci (2D prostor $f(x, y)$):

$$\hat{f}(\theta, t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y)\delta(xcos\theta + ysin\theta - t)\, dxdy \tag{9}$$

Kde:

θ je úhel, který svírá normála parametrické přímky s osou x v intervalu [$0, \pi$)
t je vzdálenost parametrické přímky od počátku
δ je Dirakova delta funkce

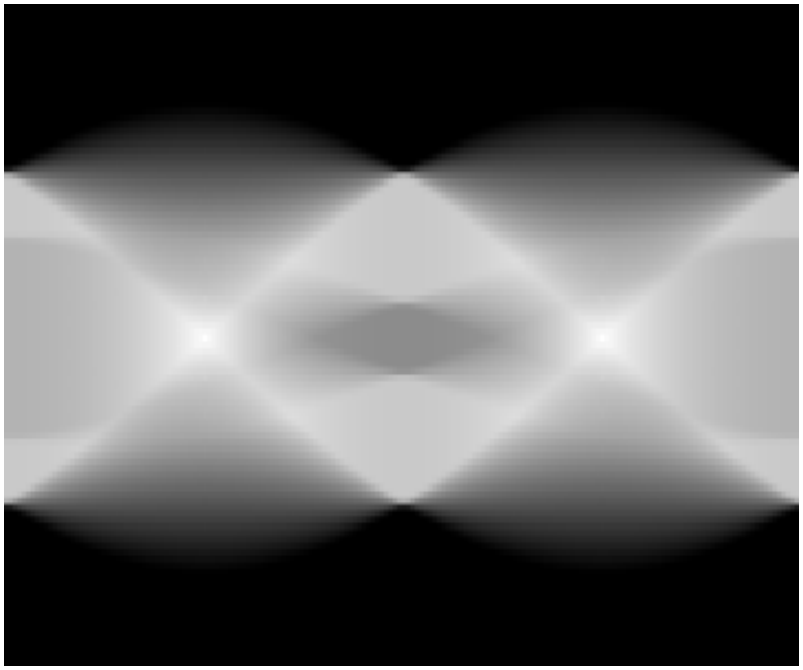
Pokud se na tuto matematickou formulaci podíváme z pohledu počítačové tomografie (CT), můžeme parametry označit jako:

θ je úhel natočení rentgenky
t je pozice rentgenky
$f(x, y)$ je zobrazovaná scéna

Výstupem transformace je matice obsahující součty intenzit podél všech přímek vedených obrazem pod všemi úhly podle θ. Výslednou matici lze vhodně vizualizovat v podobě sinogramu. Název sinogram je odvozen podle vlastnosti Radonovy transformace, kdy při provedení transformace nad Dirakovou delta funkcí vzniká sinusoida. Ukázku sinogramu můžeme vidět na obrázku 3.[1, 3]

3.3 Filtrovaná zpětná projekce

Pro výpočet inverzní Radonovy transformace v 2D prostoru existuje algoritmus, který nazýváme filtrovaná zpětná projekce. Tento algoritmus používá prosté myšlenky zpětné projekce. Průběh zpětné projekce je takový, že algoritmus vyplňuje celý prostor (rovinu) jednotlivými projekcemi obrazů pod úhlem, pod kterým vznikly. Tento postup provede



Obrázek 3: Ukázka sinogramu

pro všechny úhly a výsledné intenzity všech projekcí sečte. Vzniká tak rekonstrukce Dirakovy delta funkce. Matematicky lze zpětnou projekci zapsat formulí číslo 10:

$$p(x,y) = \int_0^{2\pi} \hat{f}(xcos\theta + ysin\theta, \theta)\, d\theta \tag{10}$$

V matematické formulaci číslo 10 můžeme vidět, že výstupem není původní funkce $f(x,y)$, jak by mohlo být očekáváno, ale $p(x,y)$. Je to z toho důvodu, že zrekonstruovaný obraz není ve skutečnosti shodný s obrazem původním. Při malém počtu zpětných projekcí vzniká okolo rekonstruovaného objektu hvězdicový artefakt, který se ztrácí s narůstajícím počtem projekcí. K potlačení vzniku artefaktu je tedy nutné provést velký počet projekcí. Další možností potlačení přítomnosti artefaktu je provedení vhodné filtrace pomocí ramp-filtrů. Při použití ramp-filtrů se stává zpětná projekce filtrovanou, odtud tedy vzniká její název samotný. Filtrace se aplikuje ještě před samotným provedením projekce. Je však nutné poznamenat, že aplikace filtrací pomocí ramp-filtrů na jednotlivé projekce vyžaduje vysoký výpočetní výkon a značně tak zpomaluje celkovou zpětnou rekonstrukci rekonstruovaného objektu. Ramp-filtry jsou obecně filtry horní propusti (high-pass), odstraňují tedy hodnoty nízkých frekvencí. Negativním jevem těchto filtrů je zesílení šumu v obrazu, který se nachází ve vysokých frekvencích. Pro odstranění (rozmazání) zmíněného šumu je nutné aplikovat další filtrace pomocí uživatelsky definovaných filtrů dolní propusti (low-pass). Výsledný filtr, který bude aplikován na projekci, vznikne konvolucí ramp-filtru a uživatelského filtru nebo jejich vynásobením ve frekvenční oblasti. Uživatelský filtr by měl být volen s ohledem na úroveň rozmazání.

V případě vysoké úrovně rozmazání dochází ke ztrátě ostrosti hran celého rekonstruovaného objektu.[2]

3.4 Praktické využití

3.4.1 Počítačová tomografie (CT)

Inverzní Radonova transformace, je intenzivně využívána ve zdravotnictví, především v počítačové tomografii (CT) k pořizování snímků průřezu vybrané části lidského těla. Počítačová tomografie využívá rentgenového záření, které umožňuje pořízení hloubkových snímků celé skenované části lidského těla, na základě vystavení těla radioaktivnímu záření. Jednotlivé vrstvy snímané části lidského těla mají rozdílnou úroveň pohlcení radioaktivních paprsků, což se na výsledném snímku rekonstruovaného objektu projeví odlišnou úrovní jasu jednotlivých vrstev. Principu rekonstrukce snímaného objektu lze využít i mimo oblast oboru počítačové tomografie (CT) a zdravotnictví. Pokud vhodně upravíme podmínky pořizování snímků, to znamená, nahradíme rentgenový zářič (rentgenku) za fotoaparát či jiné obdobné zařízení k pořizování snímků, můžeme provést snímaní a následnou rekonstrukci libovolného objektu, za podmínky dodržení základních pravidel vzdálenosti snímacího zařízení od snímaného objektu, jako je tomu v případě počítačové tomografie (CT) ve zdravotnictví. V souvislosti s nahrazením rentgenového zářiče za jiný druh snímacího zařízení s absencí radioaktivního záření, ztrácíme schopnost rekonstrukce vnitřní struktury snímaného objektu. V případě rekonstrukce objektů nevyžadujících znalost vnitřní struktury není toto omezení zásadní překážkou.

3.4.2 Spolupráce s Fakultou stavební VŠB-TU Ostrava

Na základě výše uvedené skutečnosti možnosti rekonstrukce libovolného objektu zájmu i mimo oblast zdravotnictví, vznikla myšlenka spolupráce s Fakultou stavební VŠB-TU Ostrava, k jejímu využití k rekonstrukci vybraného stavebního objektu. Následně po rekonstrukci provést měření a vyhodnocení vlastností rekonstruovaného objektu a porovnat výsledky měření s výsledky měření poskytnutými Fakultou stavební VŠB-TU Ostrava.

Jako objekty určené k rekonstrukci a měření bylo vybráno několik sloupů elektrického napětí, které svými vlastnostmi náklonu a polohy umístění poskytují vhodná data k přeměření a k následnému porovnání naměřených hodnot. Snímky sloupů byly pořízeny Fakultou stavební VŠB-TU Ostrava a současně přeměřeny pomocí standardních geodetických postupů používaných v praxi. Při tvorbě snímků byla dodržena konstantní vzdálenost snímacího zařízení od snímaného objektu. Pro každý sloup bylo dodáno celkem 8 snímků, kde jednotlivé snímky byly pořízeny pod odlišnými úhly (analogicky k pohybu rentgenky). Každý sloup byl snímán od počátečního úhlu 0 stupňů až po konečný úhel 360 stupňů s krokem 45 stupňů. Ze všech sloupů byl vybrán sloup s největším náklonem a tento sloup byl dále podroben rekonstrukci a měření s využitím inverzní Radonovy transformace. Konkrétní popis projektu spolu s výsledky měření nalezneme v kapitole 6.

4 Objemová grafika

4.1 Úvod

Pro práci s objemovým typem dat je nutné mít alespoň základní znalost teoretických poznatků, které umožní lepší pochopení dané problematiky a taktéž umožní lépe pochopit techniky jejich vizualizace. Následující kapitola si klade za cíl popsat základní teoretické principy práce s objemovými daty a jejich modelováním. V kapitole budou podrobněji vysvětleny charakteristické způsoby reprezentace objemových dat, techniky přístupu při jejich vizualizaci a taktéž problémy, které se při práci s tímto typem dat vyskytují. Rovněž bude v kapitole popsán základní matematický model, na kterém je práce s objemovými daty založena.

4.2 Obecný popis

Základní myšlenkou objemového grafiky je reprezentace objemových dat v podobě 3D modelu. Neliší se tedy od klasické 3D počítačové grafiky, která produkuje taktéž 3D modely na základě znalostí vektorů bodů v definovaném 3D prostoru. Objemová data jsou reprezentována v 3D skalárním (obecně číselném) prostoru. Původ objemových dat je často tvořen výstupem simulací a měření, které vznikají na různých specializovaných vědeckých či odborných pracovištích. Data mohou mít různé reprezentace uložení, což ovlivňuje způsob chápání jejich vizualizace. Základní princip vizualizace je založen na průchodu světelného paprsku objemem a jeho vyjádřením pomocí číselné (skalární) hodnoty. Při průchodu paprsku objemem (obecně 3D skalárním prostorem tvořícím prostředí) jsou sledovány interakce paprsku s prostředím, kde může být paprsek prostředím pohlcován nebo naopak může být prostředí dalším zdrojem záření a paprsek tak zesilovat. Po průchodu paprsku prostředím získáváme číselnou hodnotu, která popisuje výsledek interakce paprsku s prostředím. To, jak paprsek reaguje s daným prostředím, je dáno fyzikálním popisem prostředí. Pokud známe fyzikální popis prostředí, například hustotu prostředí, můžeme přizpůsobit interakci paprsku podle fyzikálního chování. Zde můžeme vidět první potenciální problém, kdy vyhodnocování interakce každého bodu objemu je výpočetně náročná úloha, která potřebuje dostatečný výpočetní výkon ke snížení časové náročnosti výpočtu. Existují však výpočetní techniky, které se snaží tento problém řešit snížením režie výpočtů při vizualizaci. Na základě znalosti získané číselné hodnoty interakce paprsku s prostředím provádíme klasifikaci této hodnoty. Klasifikací objemových dat rozumíme popsání bodů objemu tak, abychom dokázali ve výsledném obraze identifikovat, co dané body představují a přiřadili jim odpovídající průhlednost a barvu.

4.3 Matematický model

4.3.1 Spojitá interpretace

V předchozí části věnované obecnému popisu modelování objemových dat byl popsán princip šíření světelného paprsku objemem. Paprsek do objemu vstupuje v určitém bodě,

prochází objemem a poté vystupuje v koncovém bodě. Po výstupu paprsku známe hodnotu popisující interakci paprsku s prostředím v daném bodě. Přesným označením této hodnoty je intenzita nebo též radiance ve výstupním bodě. Dále je pak tato hodnota používána při klasifikaci, jak bylo uvedeno výše. Intenzita ve vstupním bodě je popsána pomocí integrálu objemového vykreslování, který je uveden ve vztahu 11:

$$I(s) = I(s_0)e^{-\int_{s_0}^{s} \kappa(t)\,dt} + \int_{s_0}^{s} q(\widetilde{s})e^{-\int_{\widetilde{s}}^{s} \kappa(t)\,dt}\,d\widetilde{s} \tag{11}$$

Uvedený vztah je složen z několika částí. Pro lepší pochopení je vhodné tento vztah rozdělit na menší části a dále je popsat samostatně. Integrál objemového vykreslování je složen z následujících částí:

- $I(s) = \boxed{I(s_0)}$

  $\boxed{I(s_0)}$ popisuje počáteční intenzitu (radianci) paprsku, ve vstupním bodem s_0. Bod s_0 je počátečním bodem, kterým vstupuje paprsek do objemu. V počátečním bodě nedochází k útlumu intenzity, znamená to tedy, že v bodě s_0 je intenzita (radiance) rovna intenzitě (radianci), se kterou paprsek vstupuje do objemu.

- $I(s) = I(s_0)\,\boxed{e^{-\tau(s_0,s)}}$
  kde $\boxed{\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)\,ds}$

  Postupným procházením paprsku skrze objem dochází k útlumu radiance. Paprsek je postupně při své cestě pohlcován objemem. τ označuje optickou hloubku mezi dvěma body. κ pak označuje absorpci světelného paprsku při postupu objemem. Paprsek objemem postupuje tak dlouho, dokud není úplně pohlcen.

  V této části končí první polovina celkového vztahu 11. Jelikož existuje možnost, že jednotlivé body objemu mohou přispívat svým světelným zářením k prostupujícímu paprsku, je nutné definovat druhou polovinu výrazu, která tento jev popisuje.

- $I(s) = I(s_0)e^{-\tau(s_0,s)} + \boxed{q(\widetilde{s})}$

  $q(\widetilde{s})$ popisuje emisi v bodě $\widetilde{s}$. Emisí rozumíme sloučení aktuální energie světelného paprsku s energií, kterou vyzařuje bod $\widetilde{s}$. Sloučením těchto energií dochází k zvýšení radiance paprsku.

- $I(s) = I(s_0)e^{-\tau(s_0,s)} + q(\widetilde{s})\,\boxed{e^{-\tau(\tilde{s},s)}}$

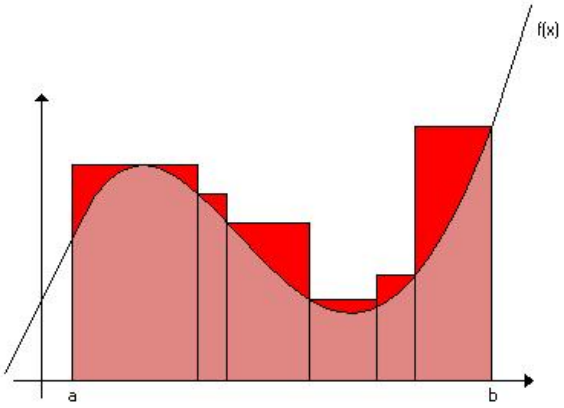
  $\boxed{e^{-\tau(\tilde{s},s)}}$ má stejný význam, jako tomu bylo v první polovině výrazu. Radiance paprsku po emisi je znovu pohlcována objemem. Paprsek pokračuje z bodu $\tilde{s}$, kde došlo k emisi a tím i navýšení radiance paprsku, dále objemem, kde je dále pohlcován až do koncového bodu s, kterým cesta paprsku končí.

- $I(s) = I(s_0)e^{-\tau(s_0,s)} + \boxed{\int_{s_0}^{s} q(\tilde{s})e^{-\tau(\tilde{s},s)}\, d\tilde{s}}$

  Protože se druhá polovina výrazu může opakovat pro více bodů, nikoliv jen pro jeden případ, je nutné provést integraci přes všechny body, které se účastní emise. Po provedení integrace získáváme celkový příspěvek všech bodů, které se podíleli na změnách intenzity (radiance) paprsku při průchodu objemem.

4.3.2 Diskrétní interpretace

V praktické realizaci je průchod paprsku objemem realizován, jako iterativní proces. Je tedy nutné upravit původní integrál objemového vykreslování do formy, která je při praktické realizaci použitelná. Tento úkol je řešen pomocí Riemannova integrálu. Hlavní myšlenkou Reimannova integrálu je rozdělení měřeného prostoru na části, které jsme schopni matematicky vyčíslit pomocí známých vzorců. Pokud se nyní omezíme pouze na 2D prostor, tak pro výpočet obsahu plochy pod grafem rozdělíme prostor na obdélníky, jejichž obsah jsme schopni snadno vyčíslit. Rozdělení je prováděno jak pro dolní, tak i pro horní odhad. Tím získáváme způsob, jak získat obsah plochy složitých objektů, pro které neznáme vzorce pro přímý výpočet. Ukázku rozdělení plochy pod grafem na obdélníky pro horní odhad lze vidět na obrázku číslo 4. Pro dolní odhad je způsob rozdělení plochy analogický k hornímu odhadu, avšak rozdíl spočívá v rozdělení plochy pouze na obdélníky, které nepřesahují křivku grafu.



Obrázek 4: Horní odhad součtu obdélníků

Po aproximaci původního vztahu 11 pomocí Riemannova integrálu získáváme vztah:

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=j+1}^{n} T_j \tag{12}$$

Kde:

$I(D)$ označuje intenzitu v koncovém bodě D
T_j označuje průhlednost v j-tém intervalu
c_i označuje intenzitu (radianci) v i-tém intervalu

Ve vztahu číslo 12 bylo zavedeno nové označení jednotlivých proměnných, které je nutné vysvětlit. Proměnná T obecně odpovídá vztahu $T_j = T(s_{j-1}, s_j) = e^{-\tau(s_{j-1}, s_j)}$, který popisuje průhlednost daného média v intervalu definovaném počátečním bodem s_{j-1} a koncovým s_j. Proměnná c_i pak značí intenzitu (radianci) v i-tém intervalu, který je dán body s_{i-1} a s_i a odpovídá tak druhé polovině výrazu číslo 11 v podobě $c_i = \int_{s_{i-1}}^{s_i} q(s)T(s, s_i)\, ds$. Přejmenování na písmeno c značí, že radianci od teď chápeme, jako barevný příspěvek v daném intervalu.

4.3.3 Kompozice

Algoritmy spadající do kategorie přímého zobrazování objemu, kterým bude později věnovaná samostatná část textu, využívají diskrétní reprezentaci, která byla uvedena v předchozím části textu. Jak již také bylo uvedeno, v diskrétním případě zobrazování objemových dat se jedná o iterativní proces, což nakonec potvrzuje i aproximovaný vzorec průchodu objemem. Kompozice je jednou z nedílných částí algoritmů přímého zobrazování objemu, jejímž účelem je sloučení všech vzorků, přes které došlo k průchodu paprskem a vytvoření tak jednotného příspěvku těchto vzorků k finální hodnotě daného bodu výsledného obrazu. Kompozice tedy spojuje aktuálně zpracovávaný vzorek se vzorky předešlými a vytváří novou hodnotu na základě alfa kompozice, která zohledňuje příspěvek průhlednosti aktuálního vzorku k naakumulované hodnotě vzorků předešlých. Barva a průhlednost jsou postupně měněny až do finální podoby, kdy paprsek končí cestu objemem. Pro celý popsaný princip slouží jednoduché kompoziční schéma, které ještě zjednodušuje diskrétní interpretaci a přibližuje ji tak více k počítačovému zpracování. Standardní kompoziční schéma při průchodu zezadu-dopředu (back-to-front) odpovídá následujícímu vztahu:

$$C_{acc} = C_{act} + (1 - \alpha_{act})C_{acc} \tag{13}$$

Kde:

C_{acc} značí dosud naakumulovanou hodnotu barvy
C_{act} značí barevný příspěvek aktuálně zpracovávaného vzorku
α_{act} značí aktuální příspěvek průhlednosti zpracovávaného vzorku

Méně častou variantou je průchod zepředu-dozadu (front-to-back), který má následující kompoziční schéma:

$$\begin{aligned} C_{acc} &= (1 - \alpha_{acc})C_{act} + C_{acc} \\ \alpha_{acc} &= (1 - \alpha_{acc})\alpha_{act} + \alpha_{acc} \end{aligned} \tag{14}$$

Kde:
C_{acc}, C_{act} a α_{act} mají stejný význam, jako ve vztahu číslo 13
α_{acc} značí dosud naakumulovanou hodnotu průhlednosti (alfa kanálu)

Informace k popisu matematického modelu ve spojité a diskrétní variantě včetně vysvětlení principu kompozice byly čerpány ze zdrojů [5, 6, 7].

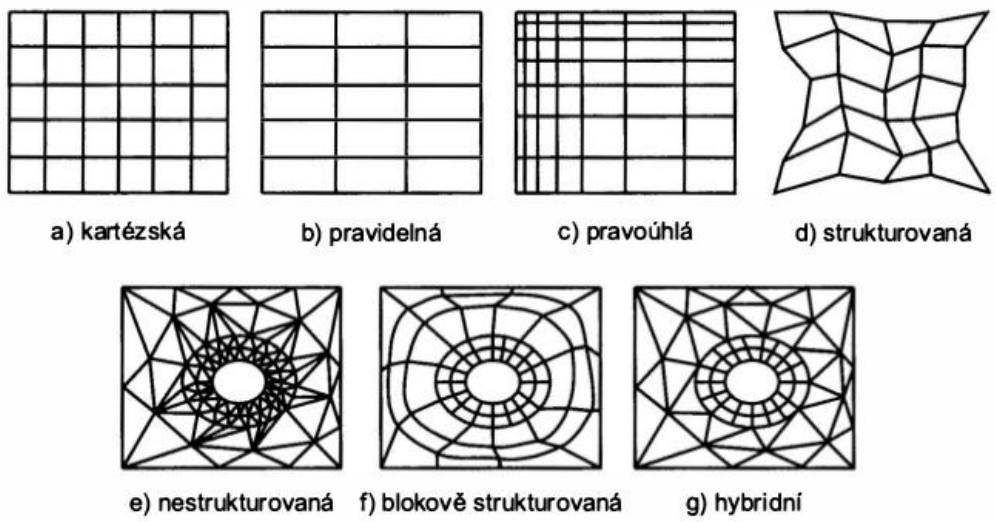
4.4 Formáty 3D dat

V úvodu byla uvedena zmínka o tom, že objemová data mají různé reprezentace uložení. Důvodem existence různých reprezentací uložení je zefektivnění práce s objemovými daty a taktéž snížení paměťové náročnosti u vybraných druhů reprezentací s využitím kompresních algoritmů. Počáteční reprezentaci dat nejsme často schopni ovlivnit, protože objemová data jsou často tvořena výstupem technických měření, medicínským výstupem či výstupem různých druhů simulací na odborných pracovištích. Základním pohledem na reprezentaci objemových dat, je pohled na jednotlivé vrstvy, které nazýváme mřížky, konkrétně na formát organizace uložení jednotlivých bodů v mřížce.[10]

Rozlišujeme těchto 7 druhů reprezentací:

1. Kartézská
2. Pravidelná
3. Pravoúhlá
4. Strukturovaná
5. Nestrukturovaná
6. Blokově strukturovaná
7. Hybridní

Pro lepší představu, jak vypadají reprezentace uložení bodů na jednotlivých mřížkách tvořících objemové struktury, je vhodné se podívat na obrázek číslo 5.



Obrázek 5: Reprezentace uložení dat v mřížce

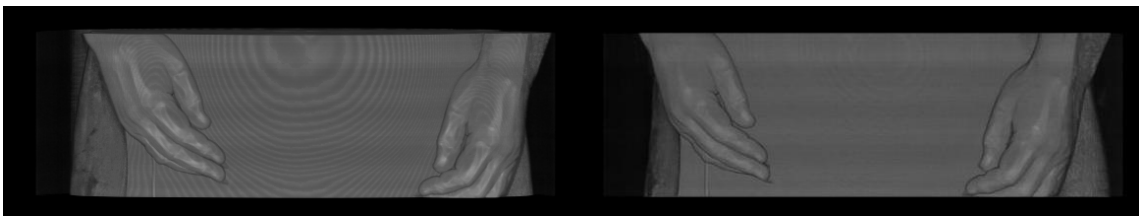
Nejčastěji se setkáváme s daty uloženými v pravidelných mřížkách a to převážně z důvodu snadného zpracování pomocí počítače. Jednotlivé body v každé mřížce ukládáme do datové struktury více rozměrného pole. V případě objemových dat se nejčastěji setkáváme s poli třetí dimenze. Neznamená to však, že jsme omezeni nejvýše třetí dimenzí. Data mohou být obecně více dimenzionální, což také není problém v případě realizace datových struktur. Objemová data se vyznačují svou datovou náročností, která se projevuje při počítačovém zpracování. Ve většině zobrazovacích algoritmů dochází k nahrání celé datové struktury nesoucí všechna objemová data do paměti počítače. Na počítače určené k vizualizaci objemových dat jsou tak kladeny vyšší výpočetní nároky. S objemovými daty lze pracovat i pomocí jiných datových struktur, než jen pomocí více dimenzionálního pole. Mezi tyto datové struktury patří varianty K-D stromů, nejčastěji pak v případě 3D dat oktalové stromy (octrees). K-D strom dělí rekurzivně prostor dimenze D na K částí, přičemž s rostoucí dimenzí dochází k exponenciálnímu růstu stupně uzlu.[8] Z popisu K-D stromu lze vidět nevýhodu v podobě rychlého růstu paměti s rychlým růstem stromu. Další nevýhoda je jev zvaný prokletí dimenzionality.[8] Tento jev se projevuje u velkých dimenzí, kde dochází k hromadění objemu na okrajích prostoru, zbytek prostoru tak zůstává nevyužit. Všechny body objemu v prostoru jsou od sebe vzdáleny ve stejné vzdálenosti.

Strukturovaná objemová data ukládáme, jako 3D pole skalárních hodnot s definovanou mřížkou rozdělení. Osm sousedních hodnot pak tvoří základní objemový element **voxel**. Voxel si můžeme představit, jako malou krychli tvořící jeden bod celkového prostoru objemu. Existují i mírně odlišné interpretace voxelu, které berou jako voxely body v průsečících mřížky, avšak práce se samostatnými voxely se v principu nemění. Voxely tedy tvoří v počítačovém podání 3D datovou strukturu reprezentující objem, která je dále iterativně procházena pomocí vizualizačních algoritmů. Zde je důležité uvést podstatný

fakt, že s takto reprezentovanými objemovými daty ve formě voxelů pracujeme s diskrétním vzorkem dat, naproti tomu původní integrál objemového vykreslování počítá s funkcí spojitou. Při průchodu voxely využívají vizualizační algoritmy vyšší frekvenci vzorkování spolu s interpolací, která definuje odpovídající bod pro dané souřadnice v objemu. Nepoužitím zvýšené frekvence vzorkování spolu s interpolací dochází ke vzniku artefaktů v rekonstruovaném obrazu. Problémům provázející rekonstrukci objemových dat bude věnována následující část textu.

4.5 Problémy rekonstrukce objemových dat

Ačkoliv reprezentace uložení objemových dat v počítačovém podání nepředstavuje zásadní překážku, jedná se pouze o počáteční část celkového procesu vizualizace. Během procesu vizualizace se můžeme setkat s problémy, které nepříznivě ovlivňují kvalitu výsledného modelu a jejichž kořeny začínají již od reprezentace dat v datových strukturách v počítači. V předchozí části textu bylo uvedeno, že objemová data v počítači představují diskrétní vzorek dat. Zde narážíme na první problém, kdy nikdy nemáme k dispozici kompletní soubor hodnot. Takto reprezentovaná data neobsahují nekonečný soubor hodnot, který je nutný pro spojitou reprezentaci, ale jen diskrétní vzorek v určitých intervalech. Úkolem vizualizačních algoritmů je sestavení původní spojité funkce, která představuje rekonstruovaný objem. Důsledkem práce s diskrétním vzorkem hodnot je stanovení ideálního počtu vzorků, které vedou ke kompletní rekonstrukci původního spojitého signálu. Tento počet lze stanovit na základě Shannonova teorému, který říká, že k rekonstrukci původního spojitého signálu potřebujeme vzorkovat s nejméně dvojnásobnou frekvencí, než je nejvyšší frekvence vyskytující se v signálu. [9] Znamená to tedy, že je nutné provést vzorkování s vyšší frekvencí (oversampling), něž kolik máme diskrétních vzorků k dispozici. V praxi většinou informaci o nejvyšší frekvenci v objemových datech nemáme k dispozici a musíme tedy určit ideální hodnotu vzorkování jiným způsobem. Mezi možné způsoby patří experimentální určení frekvence pomocí ručního zkoušení, častěji se pak volí metoda adaptivního vzorkování. Adaptivní vzorkování reaguje na velikosti mřížky a počet diskrétních vzorků a upravuje na základě těchto informací vzorkovací frekvenci. Důsledkem špatně zvolené frekvence vzorkování dochází k jevu známému jako **aliasing**. Rekonstruovaný spojitý signál je zkonstruován chybně a obsahuje tak nežádoucí obrazové artefakty. Ukázku aliasingu můžeme vidět na obrázku číslo 6.



Obrázek 6: Levá část obrazu s aliasingem, pravá část bez aliasingu

Další problém, na který lze v rámci rekonstrukce spojitého signálu narazit, je určení správné hodnoty pro body vytvořené při procesu vzorkování. Tento problém řešíme pomocí interpolace. Úkolem interpolace je určení nejvhodnější hodnoty pro daný bod vzorku na základě okolních bodů z dostupných vzorků. Interpolace také bývá součástí různých filtrů, které mohou být aplikovány pomocí operace konvoluce.

Rozlišujeme několik druhů interpolací:

- Nejbližší soused (Nearest neighbour)
- Lineární
- Bilineární
- Trilineární
- Radiální

Uvedené interpolační techniky zvyšují výpočetní náročnost vizualizačního procesu, protože jsou prováděny pro každý vzorkovaný bod. S nástupem více jádrových procesorů a moderních grafických karet je možné tento výpočet provést paralelně a snížit tak jeho časovou náročnost. Interpolace pomocí nejbližšího souseda je jednou z nejjednodušších interpolací, která je výpočetně nenáročná a je snadno implementovatelná. Její nevýhodou je však nižší kvalita interpolace, která se projevuje znatelně viditelným rastrem v rekonstruovaném modelu objemu. Lineární interpolace pracuje na principu proložení dvou bodů pomocí přímky na jedné ose. Rozšířením této interpolace do dvou souřadnicových os získáváme bilineární interpolaci, analogicky pak při rozšíření o třetí osu získáváme trilineární interpolaci. Trilineární interpolace je nejčastěji nasazovanou interpolací pro objemová data, protože poskytuje dobré výsledky s přijatelnou náročností výpočtu. Nicméně i u této interpolace může docházet k artefaktům v obraze. Radiální interpolace se pak hodí pro mřížky s nepravidelnou topologií. V praxi je nejčastěji nasazována trilineární interpolace, jak bylo uvedeno výše. Její výpočet je rozdělen do čtyř následujících částí:

1. Výpočet rozdílů mezi aktuálním bodem, předchozím bodem a následujícím bodem na všech osách (x, y, z):

$$\begin{aligned} x_d &= (x - x_0)/(x_1 - x_0) \\ y_d &= (y - y_0)/(y_1 - y_0) \\ z_d &= (z - z_0)/(z_1 - z_0) \end{aligned} \tag{15}$$

   Kde:
   x_0 značí bod přecházející před bodem x, x_1 značí následující bod po bodu x, analogicky pak platí stejně pro y_0, y_1, z_0 a z_1.

2. Provádíme interpolaci podél osy x:

$$\begin{aligned} c_{00} &= V[x_0, y_0, z_0](1 - x_d) + V[x_1, y_0, z_0]x_d \\ c_{10} &= V[x_0, y_1, z_0](1 - x_d) + V[x_1, y_1, z_0]x_d \\ c_{01} &= V[x_0, y_0, z_1](1 - x_d) + V[x_1, y_0, z_1]x_d \\ c_{11} &= V[x_0, y_1, z_1](1 - x_d) + V[x_1, y_1, z_1]x_d \end{aligned} \tag{16}$$

   Kde:
   $V[x_0, y_0, z_0]$ značí funkční hodnotu (hodnotu objemu) v bodě $[x_0, y_0, z_0]$, analogicky pak pro ostatní souřadnice.

3. Dále provádíme interpolaci podél osy y:

$$\begin{aligned} c_0 &= c_{00}(1 - y_d) + c_{10}y_d \\ c_1 &= c_{01}(1 - y_d) + c_{11}y_d \end{aligned} \tag{17}$$

4. A nakonec provádíme interpolaci podél osy z:

$$c = c_0(1 - z_d) + c_1 z_d \tag{18}$$

   Kde:
   c značí finální interpolovanou hodnotu pro daný bod objemu

Některé vizualizační algoritmy využívají k interpolaci grafickou kartu, kde k interpolaci dochází přímo v průběhu zobrazení dat. Interpolace je prováděna hardwarově pomocí grafické karty. To má velmi pozitivní důsledek na rychlost provedení interpolace. Informace k popisu interpolací byly čerpány ze zdroje [10].

4.6 Vizualizace objemových dat

4.6.1 Zobrazovací řetězec

Dosud uvedené informace o zpracování vzorků objemových dat, vzorkování a interpolaci tvoří první část celkového zobrazovacího řetězce, který je až na drobné rozdíly stejný pro všechny vizualizační algoritmy. Tento celý řetězec je složen ze šesti částí, které jsou uvedeny v následujícím seznamu podle pořadí tak, jak jsou prováděny:

1. **Vzorkování - (převzorkování viz aliasing)**
2. **Interpolace (volitelná část)**
3. **Výpočet gradientu (v závislosti na osvětlovacím modelu)**
4. **Klasifikace**
5. **Stínování (dle úrovně požadované fotorealismu)**
6. **Kompozice**

Následuje popis jednotlivých částí:

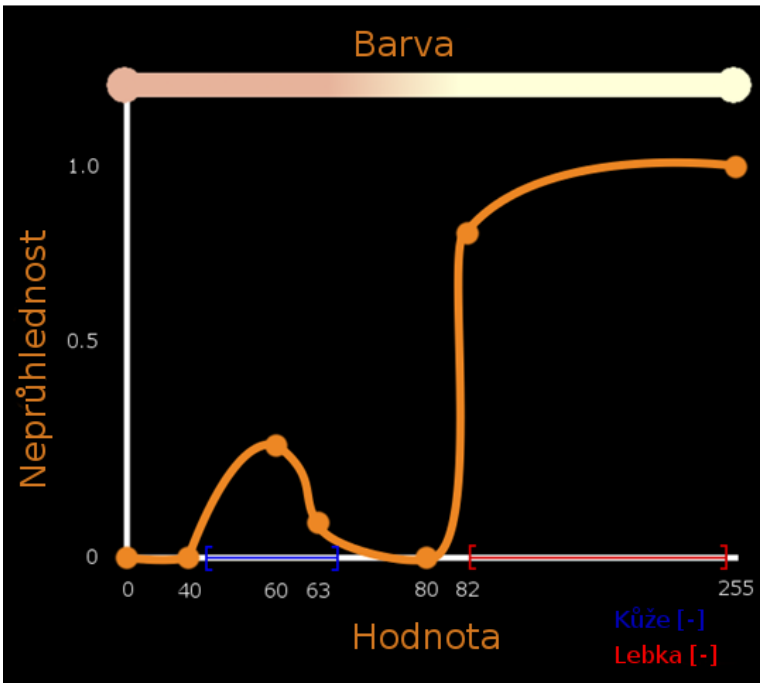
Vzorkování - vzorkování byla věnována část textu zabývající se problémy provázející vizualizaci objemových dat, viz podsekce 4.5. Způsob průchodu objemem a frekvence vzorkování jsou definovány daným algoritmem pro vizualizaci. Pokud není použito adaptivní vzorkování, je vzorkovací frekvence konstantní v celém průběhu procesu vzorkování.

Interpolace - interpolaci byla rovněž věnovaná samostatná část textu, viz podsekce 4.5. Interpolace není podmíněným krokem v celkovém procesu vizualizace objemových dat. Její absencí se však vystavujeme riziku horší kvality vizualizovaného modelu objemu.

Výpočet gradientu - vypočtená hodnota gradientu je využívána v osvětlovacích modelech, pro správné nasvícení scény s objektem. Nejčastěji je používán Phongův osvětlovací model.

Klasifikace - klasifikace je poměrně komplexní část procesu vizualizace, jejíž úlohou je namapování odpovídající průhlednosti a barvy na každý bod objemu, který spadá do určité oblasti. Oblastmi rozumíme úseky dat, ať již souvislé, což bývá častým případem nebo nesouvislé, které rozlišujeme od ostatních typů oblastí na základě konstantní veličiny, kterou je daná oblast vyplněna. Pokud tuto definici převedeme do praktické realizace modelování medicínský snímků, tak jako různé oblasti rozlišujeme typy tkání, svalovinu, kosti a další vrstvy tvořící modelovaný objekt. Pro tyto oblasti stanovujeme odpovídající průhlednost a barvu podle daného typu oblasti. Lze tak docílit barevného 3D modelu s opticky rozdílnými vrstvami. Rozhodujícím kritériem, podle kterého dochází k zařazení bodu do dané oblasti je jeho intenzita jasu (radiance). Pro tento účel

se používají tak zvané přenosové funkce, které definují příslušnou barvu a průhlednost pro danou oblast. Obecným problémem je vytvoření vhodného uživatelského rozhraní, které by usnadnilo pochopení principu přenosové funkce a jejího vhodného nastavení. Ukázku grafu přenosové funkce, který definuje jasové intervaly pro kůži a lebku a k nim odpovídající průhlednost a barvu, můžeme vidět na obrázku číslo 7.



Obrázek 7: Přenosová funkce

Stínování - stínování úzce souvisí s osvětlovacím modelem, účelem je vytvoření správně osvětleného modelu spolu se stínováním, které dodává reálnější vzhled modelu. Fotorealismus obecně není nutným požadavkem, zlepšuje však celkový dojem z modelu a prezentuje ho tak v přirozenější formě pro člověka.

Kompozice - kompozici byla věnována samostatná podsekce 4.3.3.

4.6.2 Vizualizační algoritmy

Vizualizační algoritmy vychází z matematického modelu uvedeného v podkapitole 4.3. Jejich úkolem je vyčíslení objemového integrálu a sestavení příslušného modelu objemu. Vybrané algoritmy využívají možnosti grafických primitiv k urychlení celkového procesu vizualizace, další algoritmy pak přímo implementují diskrétní interpretaci objemového integrálu.

Zobrazovací algoritmy dělíme do dvou skupin:

1. **Výpočet izoploch** - První skupinu tvoří algoritmy převádějící data do povrchové reprezentace. Při povrchové reprezentaci dochází k aproximaci povrchu viditelného objemu na síť trojúhelníků, obecně n-úhelníků. Trojúhelníky pokrývají povrch s konstantní hodnotou vzorku a vytvářejí tak síť. Vzniklé sítě nazýváme izoplochy. Odtud dostala skupina algoritmů pracujících na principu povrchové reprezentace název výpočet izoploch. Tato skupina se vyznačuje využíváním grafických primitiv poskytovaných grafickou kartou, které umožňují rychlé renderování pomocí hardwarové akcelerace. Mezi grafická primitiva řadíme body, přímky, trojúhelníky, čtyřúhelníky a obecně n-úhelníky. Vizualizovaný model nevyžaduje opakovanou rekonstrukci, lze s ním tedy velmi rychle pracovat, měnit pohledy a aplikovat různé transformace.[10]

2. **Přímé zobrazovací metody** - Algoritmy spadající do druhé skupiny pracují na principu vyčíslení integrálu objemového vykreslování. Algoritmy vzhledem k způsobu pojetí vizualizace objemu využívají grafická primitiva v omezené formě nebo vůbec. Při změně scény, aplikování transformace nebo jiné operaci s modelem, je nutné celý model opakovaně zrekonstruovat. To se negativně projevuje na výpočetní náročností celé úlohy. Avšak i přes tyto vlastnosti je skupina zastoupena vybranými algoritmy s vysokou rychlostí vizualizace s kvalitním výstupem díky rozvoji moderních grafických karet. Metody obecně poskytují více možností pro nastavení kvality výsledného modelu a poskytují tak reálnější výsledky.

Každá z uvedených skupin je zastoupena několika algoritmy. V následujícím textu budou uvedeny nejčastější algoritmy pro každou skupinu.

Neprůhledné kostky - Algoritmus řadíme do skupiny výpočtu izoploch. Algoritmus aproximuje plochu pomocí krychlí, kde u každé krychle stačí vykreslit pouze tři stěny, které jsou viditelné vůči pozorovateli. Dochází tak k značné aproximaci povrchu s nežádoucím efektem vzniku hranatého povrchu. Částečným vylepšením pomocí stínování lze dosáhnout lepšího vzhledu, hranatý povrch však zůstává.[10]

Napojování izočar - Algoritmus rovněž spadá do skupinu algoritmů výpočtu izoploch. Algoritmus pracuje s jednotlivými řezy, které mezi sebou propojuje. Na každém řezu jsou určeny body, které budou sloužit k propojení s body definovanými v následujícím řezu.

Dochází tak k aproximaci přibližného tvaru objektu v řezu například pomocí n-úhelníku. Jednotlivé vrcholy n-úhelníku jsou propojovány s příslušnými body n-úhelníků v následujícím řezu. Na výsledný model lze poté provést opláštění například namapováním textury.

Pochodující kostky (marching cubes) - Jako předchozí dva algoritmy i tento algoritmus spadá do skupiny algoritmů výpočtu izoploch. Algoritmus je nejčastějším algoritmem nasazovaným ve své skupině. V každé buňce objemu generuje algoritmus síť trojúhelníků. Buňka objemu je tvořena kostkou sestavenou z trojúhelníků. Dochází zde k využití hardwarové podpory grafické karty. Povrch je aproximován pomocí sítě malých trojúhelníků.

Vrhání paprsků (raycasting) - Tento algoritmus patří do skupiny přímých zobrazovacích metod. Algoritmus postupuje klasickým způsobem vyčíslením objemového integrálu tak, jak je uveden ve své diskrétní podobě. Prvním krokem je namapování objemových dat do krychle o rozměrech 1 x 1 x 1. Jako barvy vrcholů krychle slouží normované souřadnice definující rozměry objemu. Krychle je následně vykreslována bez předních stěn a poté bez zadních stěn. Oba způsoby vykreslení uchováváme v pomocných texturách. Tyto textury definují body vniknutí paprsku do objemu a výstupní body opuštění objemu. Barvy definované ve vrcholech slouží k identifikaci souřadnic bodů, kterými paprsek vstupuje a vystupuje.[6] Vzhledem k tomu, že známe možné body vniknutí a body opuštění objemu, lze vypočítat směrový vektor paprsku jednoduchým výpočtem:

$$direction = rayOutput - rayInput$$

Po výpočtu směrového vektoru postupuje paprsek objemem po definovaných krocích v daném směru, až na hranici objemu. Krok průchodu je definován parametrem, který je potřeba vhodně nastavit. Jeho hodnota určuje intenzitu vzorkování při průchodu paprsku objemem. Při průchodu paprsku objemem dochází k akumulaci barvy a průhlednosti podle vztahů definovaných kompozicí, viz podkapitola 4.3.3. Aby došlo k redukci kroků během průchodu paprsku objemem, je průběžně sledována akumulovaná hodnota průhlednosti a pozice paprsku v objemu. Pokud dojde k situaci, kdy je hodnota průhlednosti nízká a jedná se tedy o neprůhlednost nebo je paprsek mimo objem, je průchod předčasně ukončen. Pseudokód průchodu paprsku objemem je následující:

```
castRay(smerovy vektor)
{
    while(paprsek je v objemu)
    {
        nacti  hodnotu vzorku v objemu
        proved kompozici
        posun se o dalsi  krok v  objemu
    }

}
```

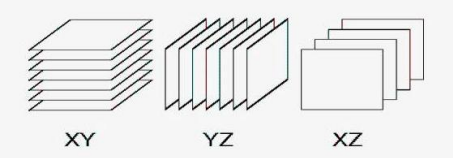
Výpis 1: Průchod paprsku objemem (RayCasting)

Jednotlivé paprsky jsou na sobě nezávislé a jejich průchody se navzájem neovlivňují. Tato vlastnost vede s nástupem více jádrových procesorů k myšlence paralelního zpracování. V jeden okamžik je možno zpracovat více paprsků současně a urychlit tak proces vizualizace. Algoritmus lze taktéž provádět paralelně na moderních grafických kartách, které umožňují provádět paralelní výpočty na samostatných výpočetních jádrech.

Texture slicing - Algoritmus je zástupcem skupiny přímých zobrazovacích metod. Patří mezi nejrychlejší algoritmy pro vizualizaci objemových dat díky značnému využití hardwarových primitiv grafické karty. Algoritmus přistupuje k objemovým datům rozdílným způsobem, než předchozí algoritmus vrhání paprsků. Objemová data můžeme chápat, jako soustavu rovnoběžných plátů umístěných na jednom zásobníku. Tato reprezentace odpovídá například snímkům z magnetické rezonance nebo CT snímkům, které tvoří soustavu snímků zachycující jednotlivé řezy snímaného tělesa. Tento zásobník plátů lze vizualizovat umístěním plátů v definovaném prostoru nad sebe a pomocí grafické karty provést kompozici průhlednosti a barvy. Jednotlivé pláty jsou reprezentovány pomocí textur a jsou namapovány na příslušná grafická primitiva, nejčastěji čtyřúhelníky. Grafická karta rovněž provádí interpolaci k dosažení vizuálně co nejlepších výsledků. Vzhledem k provádění procesu vizualizace přímo na GPU je celá operace velice rychlá.

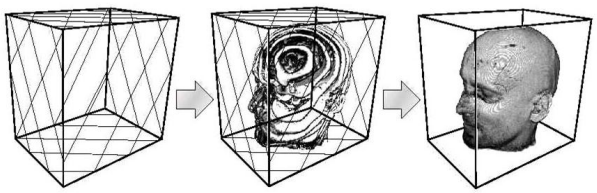
Algoritmus má dvě rozdílné techniky provedení:

- **2D texture slicing** - tato technika provedení patří mezi starší způsob realizace, který je vhodný i pro starší grafické karty nedisponující podporou 3D textur. Jednotlivé snímky jsou zarovnány do zásobníku složeného z 2D textur a tento zásobník je zobrazen v odpovídající souřadné ose proti pozorovateli. Ve skutečnosti jsou objemová data reprezentovaná v 2D pohledu, nikoliv trojrozměrně. Vzhledem k této skutečnosti je nutné uchovávat tři zásobníky se zarovnanými řezy (pláty) pro souřadné osy x, y, z. Při rotaci dochází k průběžnému přepínání mezi zásobníky podle natočení souřadných os vůči pozorovateli. Metoda využívá bilineární transformaci, která je prováděna přímo pomocí GPU k vylepšení vizuální stránky modelu. I když je metoda velmi rychlá, provází ji problém plynoucí z dvourozměrné reprezentace. Při určitém úhlu natočení dochází ke stavu, kdy jsou 2D pláty natočeny proti pozorovateli v nejednoznačné poloze. Grafická karta není schopna provést přepnutí mezi zásobníky a výsledný model není zobrazen. Přepínání mezi zásobníky také přináší negativní jev výskytu aliasingu z důvodu nedostatečné frekvence vzorkování. Počet plátů je dán počtem dostupných snímků, frekvence vzorkování je konstantní a v průběhu vizualizace ji nelze měnit. Ukázku zásobníků zarovnaných podle souřadných os lze vidět na obrázku číslo 8.



Obrázek 8: Zásobníky plátů zarovnané podle souřadných os

- **3D texture slicing** - dnes nejčastěji nasazovaná technika vizualizace objemových dat. Technika vychází z předchozí techniky 2D texture slicing, avšak zásadní rozdíl spočívá v nahrazení 2D textur pomocí 3D textur. 3D textury dnes patří k běžným funkcím, kterými jsou vybaveny současné grafické karty. U starších grafických karet nemusí být podpora 3D textur samozřejmostí a je nutné ji zkontrolovat. 3D textury jsou rozšířením klasických 2D textur, které umožňují měnit polohu v prostoru. Textury tak lze orientovat podle pozice pozorovatele. Díky této vlastnosti již není potřeba uchovávat tři kopie snímků v zásobnících, jako tomu bylo u předchozí techniky. Na 3D textury je aplikována trilineární interpolace, která znatelně zlepšuje výstupní kvalitu výsledného modelu. Vzhledem k vlastnostem 3D textur lze dynamicky upravovat frekvenci vzorkování změnou pozice textur v trojrozměrném prostoru. Lze tak měnit vzdálenost mezi texturami a frekvenci vzorkování vzhledem k aktuálnímu natočení modelu vůči pozorovateli. Trilineární interpolace je i přes zvýšenou náročnost výpočtu prováděna grafickou kartou, celý proces je tak velmi rychlý. Ukázku zobrazení objemu pomocí 3D textu lze vidět na obrázku číslo 9.



Obrázek 9: Zobrazení objemu pomocí 3D textur

Informace k vizualizační metodě objemových dat Texture slicing byly čerpány ze zdroje [11].

5 Návrh a realizace

5.1 Obecný popis realizace

V následující sekci budou popsány kroky praktické realizace modulů inverzní Radonovy transformace a vizualizace objemových dat, které vychází z teoretických východisek uvedených v předchozích kapitolách. Cílem práce je rozšíření systému FOTOM^{NG}, který je již několik let vyvíjen na Fakultě elektrotechniky a informatiky VŠB-TU Ostrava o uvedené moduly. Popis praktické realizace modulů je rozdělen do samostatných částí, které jsou věnovány konkrétní realizaci daných modulů. Dále budou popsány technické aspekty praktické realizace zahrnující použité technologie a postupy.

5.2 Modulární systém FOTOM^{NG}

Systém FOTOM^{NG} je založen na modulární platformě NetBeans Platform, na které je rovněž založeno vývojové prostředí NetBeans. NetBeans Platform je implementována v programovacím jazyku Java a je poskytována, jako základ pro tvorbu nových komplexních aplikací, které využívají možností modulárního rozšíření. [13] Vzhledem k obecným vlastnostem platformy NetBeans Platform byl vybrán, jako výchozí programovací jazyk pro tvorbu nových modulů, jazyk Java, ve kterém jsou rovněž realizovány již existující moduly systému FOTOM^{NG}.

5.3 Vývojové a testovací prostředí

Vývoj a testování modulů bylo provedeno na následující sestavě softwaru a hardwaru:

- Operační systém Windows 7 Professional 64 bit
- Vývojové prostředí NetBeans ve verzi 8.02
- JDK ve verzi 8 (32 bit)
- Procesor AMD Phenom II X4 Mobile (4 jádrový procesor, 2 GHz taktovací frekvence)
- Grafická karta ATI Mobility Radeon HD 5650 (1 GB grafické paměti GDD3)
- Operační paměť 6 GB DDR3

5.4 Modul inverzní Radonovy transfomace

Modul byl navržen na základě požadavků reálného nasazení v konkrétním projektu, který byl realizován ve spolupráci se Fakultou stavební VŠB-TU Ostrava, s cílem otestovat a porovnat výsledky měření definovaného objektu zájmu. Jako konkrétní objekt zájmu byl zvolen sloup elektrického napětí. Návrh modulu však není omezen pouze na práci s uvedeným objektem, ale je určen i pro práci s ostatními možnými objekty zájmu, které mohou být dále zkoumány. Konkrétní popis a výsledky společného projektu jsou

uvedeny v kapitole 6. Modul je složen z několika dílčích částí, které budou v následujícím textu popsány.

Základní strukturu modulu lze popsat v následujících bodech:

1. Aplikování segmentačních metod (pro všechny snímky)
2. Detekce objektu zájmu (na všech snímcích)
3. Provedení řezů objektem (na všech snímcích)
4. Provedení inverzní Radonovy transformace (na všech snímcích)

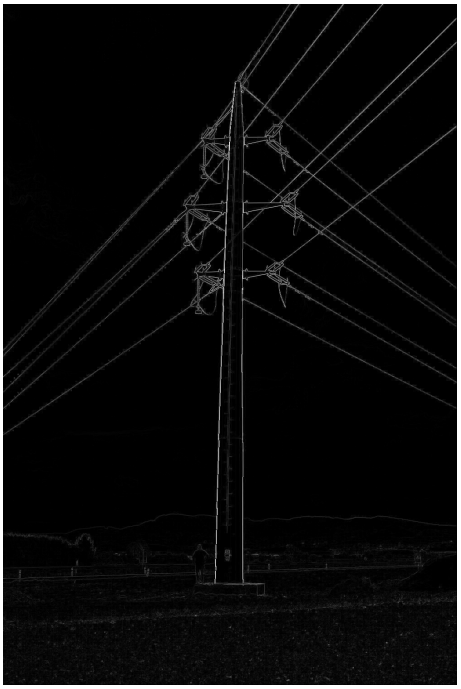
5.4.1 Segmentace obrazu

Aby bylo možné dosáhnout kvalitní rekonstrukce požadovaného objektu, vyžaduje inverzní Radonova transformace přesnou detekci řezů na snímcích. Zde narážíme na obecný problém nejen detekce řezů, ale také detekce samotného objektu zájmu na jednotlivých snímcích. V okamžiku správné detekce objektu lze omezit zpracovávanou část snímku pouze na konkrétní oblast, ve které jsou dále detekovány samotné řezy. Pro tento úkol je nutné využít možností segmentačních metod, které byly popsány v kapitole 2. V průběhu realizace modulu bylo využito reálných snímků, které byly získány v rámci spolupráce na projektu s Fakultou stavební VŠB-TU Ostrava, které obsahují typické překážky, které brání kvalitní rekonstrukci objektu zájmu. Na základě těchto překážek byly zvoleny vhodné segmentační metody, které umožnili přesnou detekci požadovaného objektu na snímcích. Ukázku snímku sloupu elektrického napětí, který spolu s ostatními snímky z dané série posloužil k návrhu modulu, můžeme vidět na obrázku číslo 10. Sloup elektrického napětí představuje objekt zájmu, na kterém jsou dále prováděny řezy. Na obrázku můžeme dále vidět nežádoucí objekty, které mohou negativně ovlivňovat detekci skutečného objektu zájmu. Mezi tyto objekty patří okolní krajina, oblast snímku s oblohou, která obsahuje souvislé oblasti mraků, projíždějící automobily a jiné nežádoucí překážky. Další překážku tvoří nerovnoměrné osvětlení scény, které vytváří nerovnoměrné rozložení jasu v oblasti hledaného objektu a znesnadňuje tak jeho přesnou detekci.

Obrázek 10: Ukázka snímku sloupu elektrického napětí

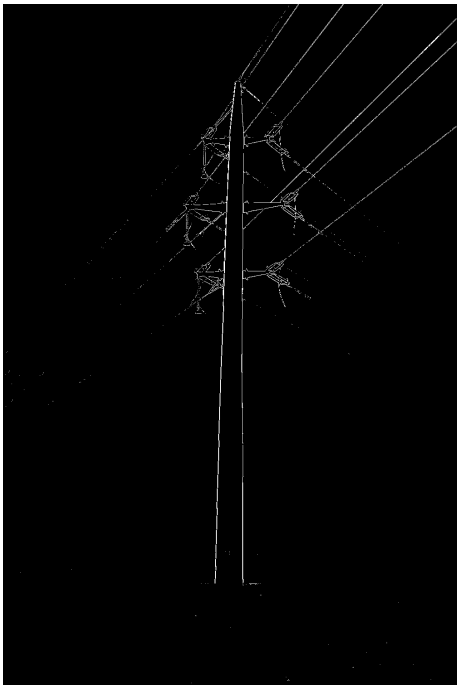
K překonání uvedených problémů byla zvolena kombinace segmentačních metod spolu s využitím funkcionality již existujícího modulu systému FOTOM^{NG}. Kombinace segmentačních metod je tvořena metodou **detekce hran**, metodou **binárního prahování** a na závěr je aplikována operace **matematické morfologie** v podobě operace uzávěru. U velmi nerovnoměrně osvětlených snímků, které vykazovali nedostatky i po aplikování uvedené kombinace segmentačních metod, bylo využito funkcionality existujícího modulu, který umožňuje ruční zvýraznění hran hledaného objektu zájmu. V následujícím textu je vhodné podrobněji popsat úlohu jednotlivých segmentačních metod. Jako první je na snímek aplikována metoda detekce hran. Úkolem metody je odstranění souvislých oblastí bodů se stejnou nebo podobnou jasovou hodnotou, které vytváří objekty ve scéně. Souvislé oblasti jsou často mylně považovány za objekty, které však nejsou cílem detekce. Typickým zástupcem, který představuje tento problém, je mrak, který je tvořen souvislou oblastí bodů. Pokud bychom se spoléhali pouze na detekci objektů na základě jasových hodnot jednotlivých pixelů, snadno bychom detekovali mnoho objektů, často by se však jednalo o vedlejší objekty, které nejsou cílem detekce. Výstupem operace je obraz obsahující hrany objektů, přičemž hledaný objekt má často hrany s vyšší jasovou hodnotou než okolní objekty. V případě, kdy je obraz nerovnoměrně osvětlen a detekované hrany nelze snadno rozlišit na základě hodnot jasu, je nutné aplikovat postup pro zvýraznění hran hledaného objektu, pomocí existujícího modulu pro úpravu snímků nebo lze využít možností externích programů pro úpravu snímků a upravené snímky zpětně načíst do systému FOTOM^{NG}. Další problém mohou představovat nežádoucí hrany, které kompli-

kují tvorbu řezů. Detekovány objekt je často tvořen hranami, které jsou sice jeho součástí, avšak v případě operace tvorby řezů představují překážky, k vůli kterým nelze snadno určit, které hrany představují hranice objektu. Řešením tohoto problému se zabývají následující části modulu. Ukázku snímku po aplikování segmentační metody detekce hran můžeme vidět na obrázku číslo 11.



Obrázek 11: Ukázka snímku po aplikování metody detekce hran

V dalším kroku je aplikována segmentační metoda binárního prahování. Úkolem metody je odstranění hran, které jsou tvořeny nízkou jasovou hodnotou a nepředstavují tak hrany hledaného objektu. Dále tento princip platí i pro nežádoucí objekty scény a hrany, které zůstaly ve scéně i po aplikování první segmentační metody. Výsledkem je obraz v binární podobě, který obsahuje pouze hrany s vyšší nebo stejnou jasovou hodnotou, jako je stanovená hodnota prahu. Ukázku snímku po aplikování segmentační metody binárního prahování můžeme vidět na obrázku číslo 12.



Obrázek 12: Ukázka snímku po aplikování metody binárního prahování

I po aplikování druhé segmentační metody nelze vyloučit situaci, že výsledný binární obraz bude obsahovat nežádoucí hrany. Tento stav lze částečně eliminovat nastavením vyšší prahové hodnoty, avšak za cenu rizika možného přerušení hran, které chceme zachovat. Aby bylo možné použít vyšší prahové hodnoty a současně nepřijít o potřebné detaily objektu, je operace binárního prahování automaticky následována operací uzávěru matematické morfologie. Operace uzávěru opravuje místa, kde došlo vlivem prahování k přerušení hran. Dále vyplňuje prázdná místa, která mohou vzniknout uvnitř objektů. V kombinaci s metodou binárního prahování lze dosáhnout vyšší variability při volbě prahové hodnoty.

Uvedené segmentační metody jsou aplikovány na všechny snímky z dané série. Uživateli je umožněno nastavit hodnotu prahu pro jednotlivé snímky individuálně. Uživatel také může aplikovat uvedené metody jednotlivě na vybraných snímcích, s cílem dosažení nastavení co nejpřesnějších hodnot parametrů segmentace.

5.4.2 Detekce objektu zájmu

V této části modulu dochází k detekci samotného objektu zájmu, na kterém jsou následně prováděny horizontální řezy. Úspěšnost operace je závislá na předchozích krocích segmentace obrazu. Pokud byla segmentace provedena s vhodnými parametry pro všechny snímky, lze stanovit hranice objektu a provést řezy. V návrhu aplikace je počítáno s horizontálními řezy, které jsou vzhledem k povaze rekonstruovaných objektů

vhodnější. V předchozí části textu věnované segmentaci bylo uvedeno, že i po aplikování obou segmentačních metod zůstávají v obrazu doplňkové hrany, které nejsou přínosné pro přesnou lokalizaci objektu. Tyto hrany nyní komplikují přesnou detekci objektu a je nutné je eliminovat. Uvedený problém je v modulu řešen pomocí uživatelsky definované oblasti výběru v aktuálně zobrazeném snímku, ve kterém se nachází objekt zájmu. Výběr je následně aplikován na ostatní snímky z dané série. Aplikování výběru na zbylé snímky je umožněno na základě předpokladu konstantní polohy snímaného objektu na všech snímcích, který musí být dodržen při pořizování snímků snímacím zařízením. Výběr umožňuje pracovat i s malou odchylkou polohy objektu na jednotlivých snímcích, avšak při velké odchylce je detekce méně přesná a může docházet k nerozpoznání hranic objektu. Po aplikování výběru získáváme dostatečně přesnou polohu objektu zájmu na všech snímcích a současně eliminujeme nežádoucí vnější hrany. Získaná oblast je dále použita při tvorbě řezů, které jsou vytvářený v následujícím kroku.

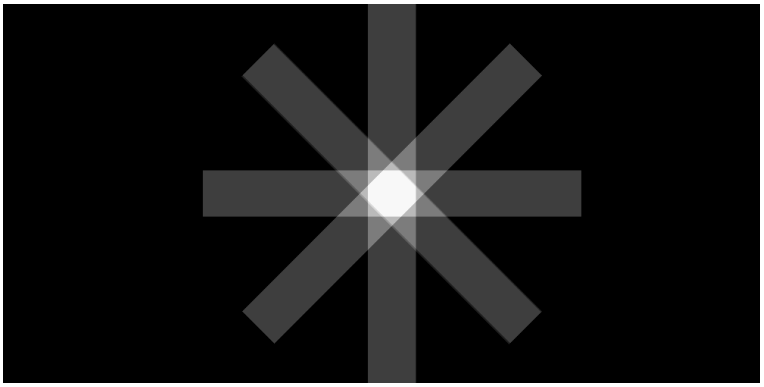
5.4.3 Provedení řezů

Provádění řezů objektem probíhá v oblasti získané z předchozí části modulu. Každý horizontální řez je tvořen úsečkou, která je definována počátečním a koncovým bodem. Tyto body jsou určeny hranami objektu, které v této fázi známe. Algoritmus tvorby řezu postupuje bod po bodu, až do momentu, kdy je nalezena levá hrana objektu zájmu. Po nalezení levé hrany je definován počáteční bod úsečky a je pokračováno hledáním koncového bodu úsečky. Algoritmus pokračuje dále bod po bodu, až do nalezení pravé hrany, která definuje polohu koncového bodu úsečky. Algoritmus pracuje s minimální délkou řezu, která zabraňuje uváznutí algoritmu v případě příliš krátké hrany objektu. Počet řezů je stanoven na základě uživatelského vstupu. Zvolený počet řezů je poté aplikován na všechny snímky z dané série. Získáváme tak řezy určené polohou v objektu a úhlem, pod kterým byl nasnímán objekt na daném snímku. Získané řezy představují vstup pro inverzní Radonovu transformaci. Ukázku uživatelského výběru spolu s provedenými řezy můžeme vidět na obrázku číslo 13. Ve vybraných případech můžeme narazit na problém, kdy délka řezu přesáhne koncovou levou hranu a pokračuje dále až do hranice definované oblastí výběru. Tento problém nastává v případě příliš vysoké prahové hodnoty, která byla použita v druhé fázi segmentace a přerušila tak hranu objektu. Problém je možné řešit dvěma způsoby. První způsob vyžaduje změnu prahové hodnoty na nižší úroveň, která zesílí hrany objektu a poté opětovné provedení řezání objektu. Druhý způsob je založen na ruční úpravě problematických řezů. Modul uživateli umožňuje vybrat konkrétní řezy a tyto řezy přesunout o jeden či více bodů výše nebo níže na místa, kde jsou hrany objektu nepřerušené. Jednotlivé řezy je nutné zkontrolovat a případně upravit na všech snímcích z dané série. Pokud řezy, které na sebe v jednotlivých snímcích navazují, budou nepravidelné délky, bude výstup rekonstrukce pomocí inverzní Radonovy transformace deformován.

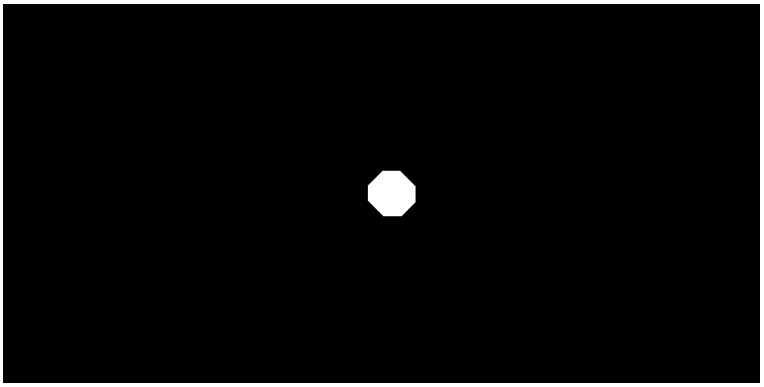
Obrázek 13: Ukázka uživatelského výběru spolu s řezy

5.4.4 Provedení inverzní Radonovy transformace

Posledním krokem, který je v navrženém modulu vykonán, je provedení inverzní Radonovy transformace pro všechny řezy na všech snímcích z daná série. Úspěšnost rekonstrukce je dána kvalitou výstupů z předchozích kroků. Inverzní transformaci lze také provést pro jednotlivé řezy, které si zvolí uživatel zvlášť. Výstupem transformace pro každý řez je snímek rekonstruovaného objektu spolu s hvězdicovým artefaktem, který byl popsán v kapitole 3. Druhou část výstupu tvoří filtrovaný snímek bez hvězdicového artefaktu, který obsahuje pouze rekonstruovaný objekt. Uvedené snímky můžeme vidět na obrázcích 14 a 15. Pozice rekonstruovaného objektu na snímku odpovídá pozicím jednotlivých řezů v původních snímcích, ze kterých byl objekt pomocí inverzní transformace zrekonstruován. Tato vlastnost umožňuje s výstupem dále pracovat pomocí vybraných rekonstrukčních technik, které nám umožňují provádět rekonstrukci a měření objektu zájmu. Jednou z technik, která umožňuje z dostatečného počtu řezů zrekonstruovat původní objekt v 3D podobě, je technika rekonstrukce pomocí objemové (voxelové) grafiky.



Obrázek 14: Rekonstruovaný objekt s hvězdicovým artefaktem



Obrázek 15: Rekonstruovaný objekt bez hvězdicového artefaktu

5.5 Modul vizualizace objemové (voxelové) grafiky

Druhou část praktické realizace tvoří modul pro vizualizaci objemové (též voxelové) grafiky. Popisu zpracování a vizualizaci objemových dat byla věnována rozsáhlá kapitola 4. Navržený modul implementuje dva zobrazovací algoritmy, mezi kterými lze uživatelsky přepínat. Implementované zobrazovací algoritmy tvoří algoritmus RayCasting a algoritmus 3D Texture Slicing. Algoritmy byly zvoleny na základě jejich vlastností, které byly podrobněji popsány v kapitole 4. V rámci implementace modulu bylo využito vybraných technologií, které umožnily zvýšit celkový výkon aplikace. Popis použitých technologií spolu s popisem implementace modulu bude uveden v následujícím textu.

5.5.1 Použité technologie

Práce s objemovými daty a jejich vizualizace vyžaduje efektivní využití dostupného výpočetního výkonu počítače, pro dosažení optimálního chodu aplikace. Pro platformu Java existuje více možností, jak navýšit výkon aplikace. Základní technika, která je používána současnými virtuálními stroji platformy Java k navýšení rychlosti často prováděného kódu, se nazývá Just In Time kompilace, zkráceně (JIT). JIT překlad je speciální technika, která překládá spuštěný program do strojového kódu počítače. Technika zvyšuje celkový výkon aplikace, avšak nevýhoda spočívá v časové prodlevě, ke které dochází před spuštěním samotného překladu. Další technikou je implementace kritických částí kódu v nativním jazyku (například C/C++) a poté volání takto implementovaných částí kódu z jazyku Java. Technika je v programovacím jazyku Java známa pod označením JNI (Java Native Interface). JNI je používána vybranými knihovnami k dosažení maximálního výkonu prováděného kódu. V současnosti je také často nasazována technika paralelního vykonávání kódu, která je díky rozšíření více jádrových procesorů snáze realizovatelná. Omezení představují algoritmy, které nejsou plně paralelizovatelné.

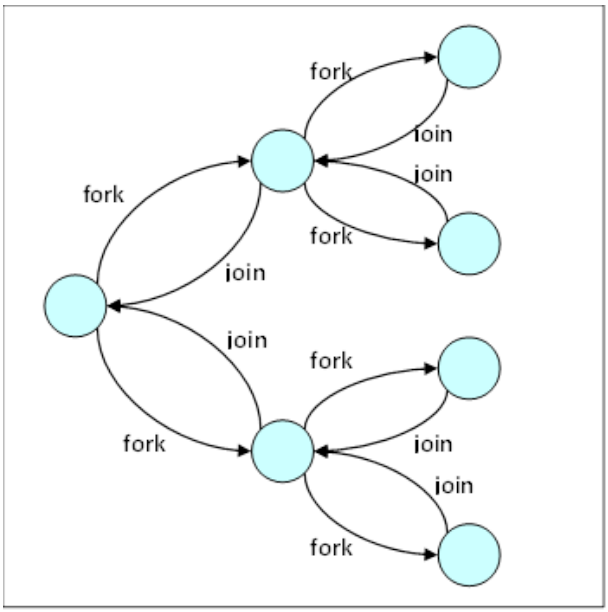
Navržený modul využívá kombinaci uvedených optimalizací. V následujícím výpisu jsou uvedeny technologie, které byly použity v rámci praktické realizace modulu.

Výpis použitých technologií:

1. Grafická knihovna JOGL ve verzi 2.1
2. Fork Join Framework - framework pro paralelní vykonávání kódu

Grafická knihovna JOGL - Použitá grafická knihovna slouží, jako přímá nadstavba (anglicky wrapper) nad knihovnou OpenGL. Knihovna je implementovaná technikou JNI a volá tak přímo metody implementované v nativním jazyku C. Důvodem nasazení knihovny je umožnění přímého využití grafické karty, převážně hardwarově akcelerovaných grafických primitiv. Nativní knihovna OpenGL, nad kterou je postavena knihovna JOGL, je multiplatformní. Díky této vlastnosti lze implementovaný modul použít nad operačními systémy MS Windows, Linux a MAC OS.[12]

Fork Join Framework - Jedná se o rozšíření standardní kolekce tříd paralelního programování v jazyku Java, které přidává nové techniky paralelizace. Toto rozšíření je standardní součástí JRE (JDK) od verze 7. Jednou z nově přidaných technik je fork-join model. Principem fork-join modelu je rekurzivní rozdělení dílčí úlohy na menší části (fork), které jsou následně paralelně zpracovávány a poté jsou výsledky zpracování zpětně spojeny (join) v jeden celek. Schéma fork-join modelu můžeme vidět na obrázku číslo 16.



Obrázek 16: Schéma fork-join modelu

Ve stavu, kdy je dílčí úloha rekurzivně rozdělena na menší části, je nutné zvolit algoritmus, který tyto jednotlivé části zpracuje. Fork Join Framework řeší tento úkol pomocí algoritmu work stealing.[14] Algoritmus je založen na principu spolupráce vláken, které se podílí na zpracování jednotlivých částí. Každé vlákno obsahuje frontu úkolů, které má v plánu obsloužit. Úkoly rozumíme jednotlivé části, které vznikly rozdělením původní úlohy. Vlákna poté zpracovávají své fronty úkolů a v případě, kdy kterékoliv vlákno vyprázdní svou frontu, převezme několik úkolů z front ostatních vláken, které stále nedokončili své úkoly, a pokračuje dále v práci. Tento princip platí navzájem pro všechny vlákna. Snahou je co nejrychlejší paralelní zpracování všech úkolů.

Fork Join framework v navrženém modulu slouží k paralelnímu zpracování snímků, které tvoří objemová data. Série snímků představují vhodná data k paralelnímu zpracování, vzhledem k možnosti rozdělení na části, které lze nezávisle na sobě zpracovávat. Druhým důležitým využitím Fork Join Frameworku byla implementace algoritmu RayCasting, který byl podrobněji popsán v podsekci 4.6.2. Algoritmus vrhá paprsky objemem nezávisle na sobě, proto lze tyto paprsky paralelně zpracovat a docílit tak rychlejší vymodelování výsledného 3D objektu.

5.5.2 Algoritmus 3D Texture Slicing

Algoritmus 3D Texture Slicing představuje první z implementovaných algoritmů, které jsou v modulu použity pro vizualizaci objemových dat. Základní prvek algoritmu představuje 3D textura, jak již samotný název napovídá. Tato textura je realizována pomocí grafické knihovny JOGL, která umožňuje vytvořit 3D texturu přímo pomocí grafické karty. Pro dosažení vizuálně dobrého výsledku je na grafické kartě hardwarově prováděna trilineární interpolace. Implementovaný algoritmus představuje jen jednu z částí zobrazovacího řetězce, přes který jsou objemová data zpracovávána, až do konečné podoby na obrazovce.

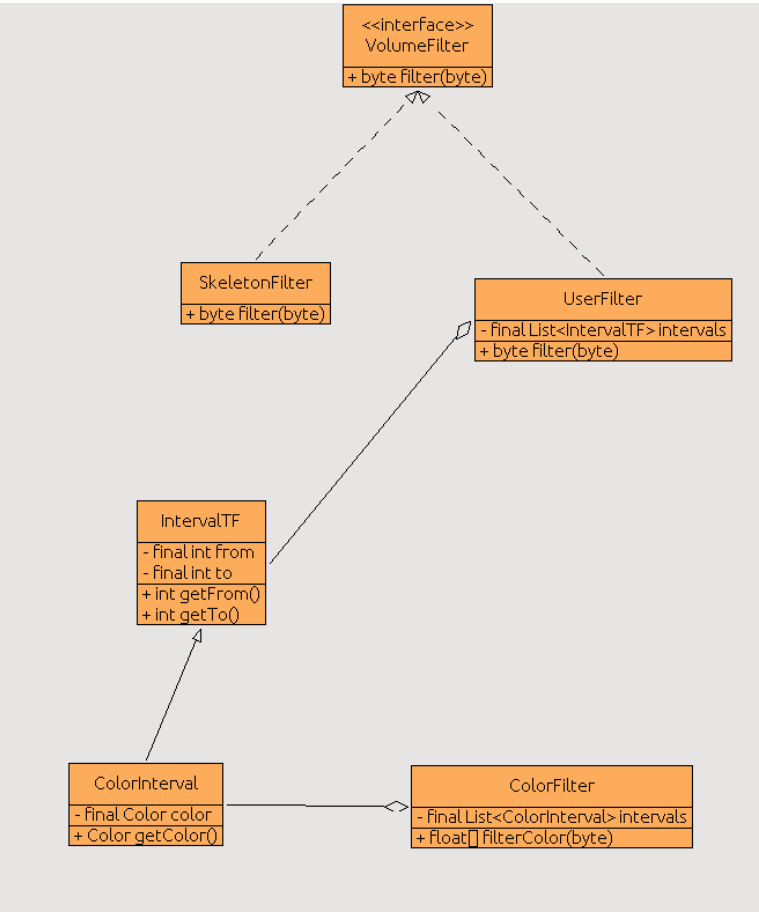
Základní části zobrazovacího řetězce:

1. Načtení objemových dat
2. Provedení filtrace jasu
3. Provedení klasifikace (namapování barvy)
4. Vizualizace pomocí algoritmu 3D Texture Slicing

Načtení objemových dat - Objemová data jsou načítána v podobě série snímků, které tvoří průřezy snímaným tělesem. Tato reprezentace odpovídá snímkům z počítačové tomografie (CT), které jsou rovněž rozděleny do série. Po načtení dat z jednotlivých souborů snímků, jsou data převáděna do datové struktury pole bytů, které představuje úložiště jednotlivých voxelů. Vytvořené pole je dále předáno dalším částem zobrazovacího řetězce ke zpracování. Modul není omezen pouze na lékařské snímky. Pokud je dodána jako vstup série snímků, které v pravidelných řezech mapují libovolný objekt, jsou tyto snímky dále předány k vizualizaci. Podporované formáty vstupních dat jsou obrazové soubory ve formátu JPEG, PNG, BMP a GIF. Důležitým faktorem je název jednotlivých souborů snímků. Snímky by měly být pojmenovány abecedně tak, aby bylo dodrženo pořadí, jak snímky na sebe navazují. Předchozí modul inverzní Radonovy transformace pojmenovává výstupní snímky číselným označením od 0 do konečného počtu. V případě, kdy není dodrženo vhodné pojmenování, jsou soubory načítány v nepravidelném pořadí. Výsledkem je poté deformovaný objekt, který byl z těchto dat zrekonstruován.

Provedení filtrace jasu - Dalším krokem je filtrace objemových dat. Pokud uživatel nastaví filtr, který může být vybrán z dostupných filtrů obsažených v modulu nebo filtr přímo vytvoří, je prováděna filtrace objemových dat. Úkolem filtrace je odstranění jasových hodnot, které nespadají do definovaného intervalu přípustných hodnot filtrace. Základním scénářem použití filtrace je zvýrazněné kostí u CT snímků lidského těla. Body kostí představují obecně vyšší jasové hodnoty, než jasové hodnoty tkání. Filtr potlačí nízké jasové hodnoty a vysoké ponechá. Výsledkem je zobrazení kostí bez tkaní. Filtry lze uživatelsky definovat pomocí nastavení jasových intervalů bodů, které mají být ponechány. Tímto způsobem lze dynamicky přizpůsobovat filtraci vzhledem k aktuálnímu typu objemových dat. Objemová data představují rozsáhlou datovou strukturu,

kde je nutné filtraci provést pro všechny body objemu. Modul zde využívá Fork-Join Frameworku k paralelnímu zpracování objemových dat. Data jsou rozdělena na menší části, které jsou paralelně zpracovány. V případě složitějších filtrů, které obsahují více intervalů jasových hodnot, dochází k znatelnému zrychlení filtrace. Každý filtr, který je v modulu obsažen, implementuje rozhraní, které je dále používáno vizualizačními třídami. Úkolem rozhraní je sjednocení způsobu komunikace mezi filtry a samotnými vizualizačními třídami. Uvedený způsob implementace umožňuje budoucí přidání nových filtrů, bez nutností změny vnitřní struktury modulu. Ukázku třídního diagramu popisujícího způsob implementace filtrů můžeme vidět na obrázku číslo 17.



Obrázek 17: Třídní diagram popisující implementaci filtrů objemových dat

Z obrázku číslo 17 můžeme vidět, že k přidání nového filtru objemových dat stačí implementovat pouze rozhraní VolumeFilter, které deklaruje jednu metodu.

Provedení klasifikace (namapování barvy) - Následující krok úzce navazuje na krok předchozí. Stejně jako předchozí krok je i tento krok volitelný. K aplikování dochází v případě definování klasifikačního filtru. Filtr je založen na stejném principu, jako před-

chozí jasový filtr, avšak nedochází zde k potlačení jasových hodnot, ale k namapování zvolené barvy na body v daném jasovém intervalu. Vybrané oblasti objektu, které se nachází ve stejné vrstvě objemu, mají obvykle konstantní jasovou hodnotu. Příkladem může být vrstva kůže lidského těla, které má vyšší jasovou hodnotu bodů, než vrstva tkání, ale naopak nižší než vrstva kostí. Tímto způsobem lze provést klasifikaci jednotlivých vrstev objektu a zobrazit tak vizualizovaný objekt v barevném podání. I klasifikace probíhá paralelně pomocí Fork-Join Frameworku. Zpracování jednotlivých částí objemových dat je na sobě nezávislé. Díky této vlastnosti zde nedochází ke kolizím při zpracování jednotlivých částí dat a jednotlivé filtry jsou tak paralelně aplikovány.

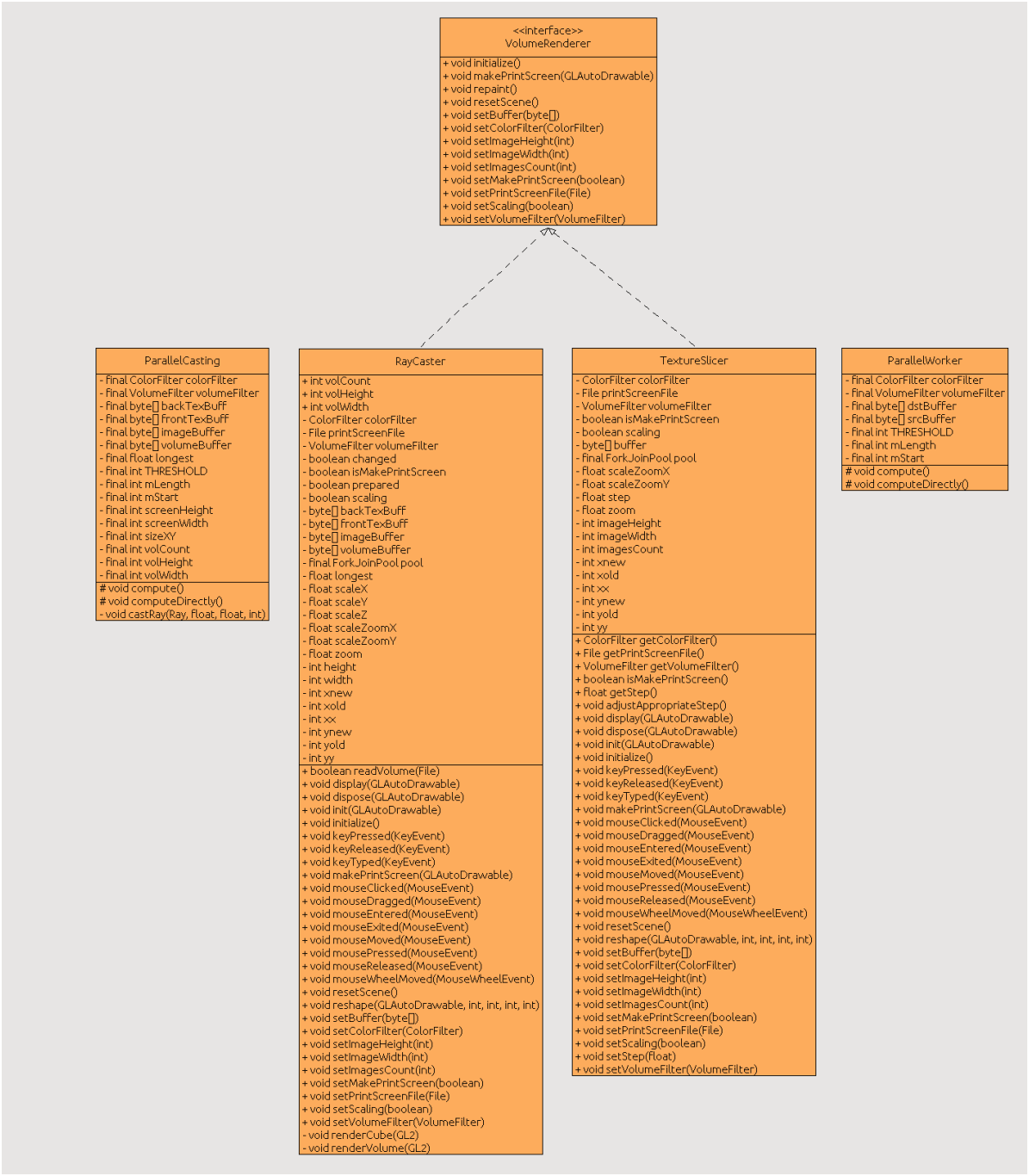
Vizualizace pomocí algoritmu 3D Texture Slicing - Posledním krok zobrazovací řetězce tvoří samotná vizualizace pomocí algoritmu 3D Texture Slicing. Detailní popis algoritmu byl uveden v sekci 4.6.2. Vzhledem k plánovanému nasazení dvou vizualizačních algoritmů byla funkcionalita obou algoritmů sjednocena pomocí jednotného rozhraní, které umožňuje pracovat s více vizualizačními algoritmy. Cílem rozhraní je sjednocení komunikace mezi modulem a třídami implementující toto rozhraní. Modul je tak otevřen budoucí možnosti přidání nových vizualizačních algoritmů, bez nutnosti změny vnitřní logiky. Ukázku třídního diagramu popisujícího strukturu dostupných vizualizačních tříd můžeme vidět na obrázku číslo 18.

Vizualizační třída, která implementuje uvedený algoritmus, očekává objemová data, která již byla zpracována pomocí předchozích kroků filtrace. Přijatá data následně vizualizuje pomocí 3D textur. Ukázka části kódu provádějící vizualizaci pomocí 3D textur můžeme vidět ve výpisu číslo 2.

```
for(float fIndx = −1.0f; fIndx <= 1.0f; fIndx += step) {
    gl.glBegin(GL_QUADS);
     float zCor = fIndx;
     float texIndex = (zCor + 1.0f) ∗ 0.5f;
     gl.glTexCoord3f(0.0f, 0.0f, texIndex);
     gl.glVertex3f(−1.0f, −1.0f, zCor);
     gl.glTexCoord3f(1.0f, 0.0f, texIndex);
     gl.glVertex3f(1.0f, −1.0f, zCor);
     gl.glTexCoord3f(1.0f, 1.0f, texIndex);
     gl.glVertex3f(1.0f, 1.0f, zCor);
     gl.glTexCoord3f(0.0f, 1.0f, texIndex);
     gl.glVertex3f(−1.0f, 1.0f, zCor);
     gl.glEnd();
}
```

Výpis 2: 3D Texture Slicing - vizualizace

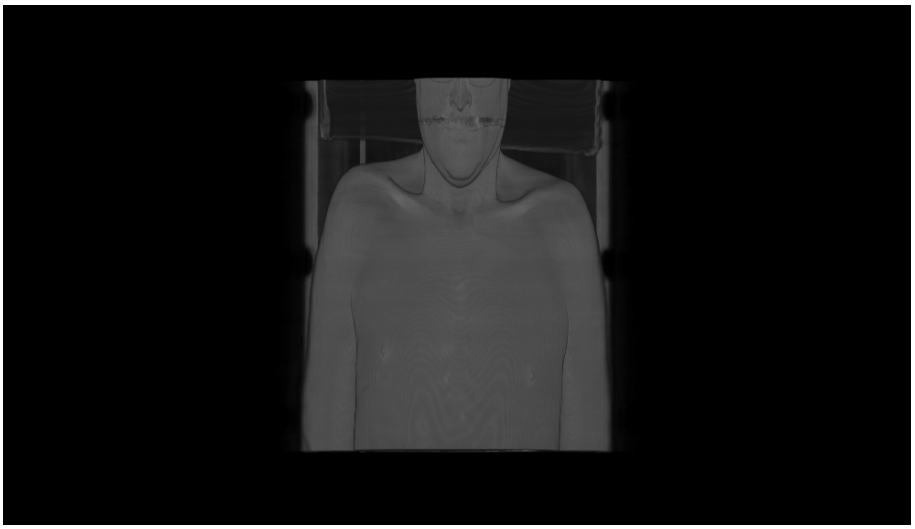
Z výpisu můžeme vidět, že každá 3D textura je mapována na čtyřúhelník umístěný v ortogonálním prostoru, který je definován intervalem (-1, 1) v souřadných osách x, y, z. Samotné 3D textury jsou v grafické paměti umístěny v krychli s délkou hrany 1. Je tedy nutné přepočítat souřadnice 3D textury do vhodného intervalu (interval 0 až 1). Tento výpočet znázorňuje inicializace proměnné s názvem $texIndex$. Proměnná s názvem step



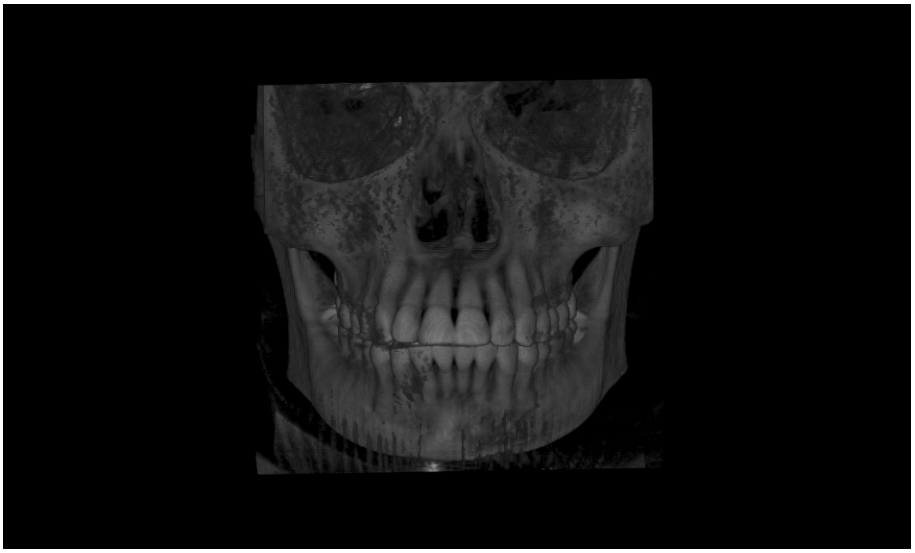
Obrázek 18: Třídní diagram popisující způsob implementace vizualizačních tříd

určuje úroveň vzorkování. Tato hodnota je automaticky upravována na základě vlastností objemových dat, která jsou aktuálně vizualizována. Vzhledem k principu vizualizace, která využívá grafická primitiva včetně samotných 3D textur, jsou uživatelské transformace, jako je například rotace, velmi rychlé a umožňují reagovat na tyto události v reálném čase.

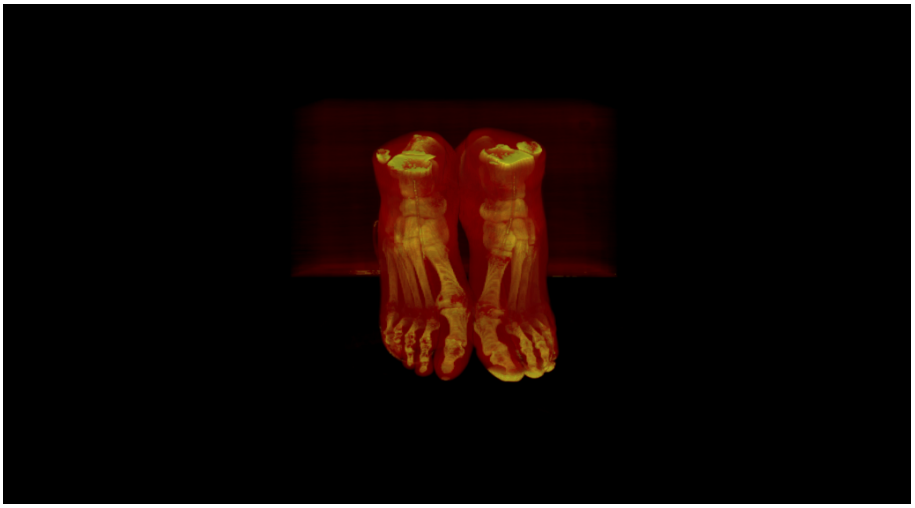
Ukázka výstupu vymodelovaných objektů pomocí algoritmu 3D Texture Slicing:



Obrázek 19: 3D model hrudníku



Obrázek 20: 3D model lebky



Obrázek 21: 3D model chodidel v barevném podání

5.5.3 Algoritmus RayCasting

Algoritmus RayCasting je druhým zástupcem z dostupných vizualizačních algoritmů. Algoritmus rovněž využívá funkcionality grafické knihovny JOGL, avšak princip zobrazení objemových dat je odlišný od principu algoritmu 3D Texture Slicing. Algoritmus byl podrobně popsán v podsekci 4.6.2. Hlavní rozdíl spočívá v postupu vizualizace objemových dat. Jako první krok je vytvořena krychle o délce hrany 1, jejíž stěny jsou vyplněny barevnými přechody, které jsou vypočítány na základě normovaných hodnot barev ve vrcholech. Normované hodnoty barev ve vrcholech jsou stanoveny na základě rozměrů objemových dat. Tento krok je realizován pomocí grafické knihovny JOGL, ze které jsou využity pro vytvoření krychle dostupná grafická primitiva. V následujícím kroku je krychle zobrazena bez předních stěn a následně bez zadních stěn. Tímto krokem definujeme počáteční a koncové body, kterými parsek do objemu vstupuje a poté vystupuje. Zobrazenou krychli v obou pohledech ukládáme do pomocných datových struktur, které jsou následně použity v procesu vizualizace. Od tohoto kroku již algoritmus nevyužívá žádná grafická primitiva a celý výpočet se přesouvá na dostupnou výpočetní jednotku, v aktuální implementaci na dostupný procesor. Rychlost vizualizace je značně závislá na dostupném výpočetním výkonu. Algoritmus je ze své povahy paralelizovatelný, tudíž při jeho běhu představují výhodu výpočetní jednotky s vysokým počtem jader, které jsou schopny paralelního běhu. Modul využívá Fork-Join Frameworku k rozdělení výpočetní úlohy na dostupná výpočetní jádra s cílem využití maximálního paralelního výkonu při procesu vizualizace. Zobrazovací řetězec je tvořen stejnými kroky jako je tomu u algoritmu 3D Texture Slicing. Rozdílný je pouze čtvrtý krok, kde je prováděna vizualizace pomocí algoritmu RayCasting.

Upravený řetězec odpovídá následující podobě:

1. Načtení objemových dat
2. Provedení filtrace jasu
3. Provedení klasifikace (namapování barvy)
4. Vizualizace pomocí algoritmu RayCasting

Vizualizace pomocí algoritmu RayCasting - První tři části zobrazovacího řetězce, které jsou shodné, byly popsány v předchozí části textu věnované algoritmu 3D Texture Slicing. Čtvrtá část je jedinou částí zobrazovacího řetězce, která je odlišná od předchozího případu. Stejně, jako v předchozím případu, je i zde komunikace s vizualizačním algoritmem zajištěna přes jednotné rozhraní, které je implementováno vizualizační třídou, viz obrázek číslo 18. Po průchodu prvními třemi částmi zobrazovacího řetězce jsou objemová data předána vizualizační třídě. Následně algoritmus paralelně simuluje průchod paprsků objemem a vyčísluje objemový integrál, který byl popsán v sekci 4.3. Ukázku části kódu simulující průchod paprsků objemem, která je paralelně vykonávána, můžeme vidět ve výpisu číslo 3.

```
protected void computeDirectly() {
    int y = mStart / screenWidth;
    int x = mStart - (y * screenWidth);
    for (int i = mStart; i < mStart + mLength; i++, x++) {
        if (x >= screenWidth) {
            x = 0;
            y++;
        }

        if (y >= screenHeight) {
            return;
        }

        int i0 = (y * screenWidth + x) * 4;
        int i1 = i0 + 1;
        int i2 = i0 + 2;

        Point3D backHit = new Point3D(new Vector3f((float) (backTexBuff[i0] & 0xFF) / 256.0f, (float
            ) (backTexBuff[i1] & 0xFF) / 256.0f, (float) (backTexBuff[i2] & 0xFF) / 256.0f));
        Point3D frontHit = new Point3D(new Vector3f((float) (frontTexBuff[i0] & 0xFF) / 256.0f, (
            float) (frontTexBuff[i1] & 0xFF) / 256.0f, (float) (frontTexBuff[i2] & 0xFF) / 256.0f))
            ;

        backHit.mulPointOperation(longest - 1.0f);
        frontHit.mulPointOperation(longest - 1.0f);

        Vector3f dir = backHit.subOperation(frontHit);

        float sqrLen = dir.sqrMag();
        if (sqrLen > 0.0001f) {
            float len = (float) Math.sqrt(sqrLen);
            dir = dir.mulOperation(1.0f / len);
            float stepSize = 0.5f;
            castRay(new Ray(frontHit, dir), stepSize, len, i0);
        }
    }
}
```

Výpis 3: RayCasting - vizualizace

V uvedeném výpisu můžeme vidět část kódu, která se stará o simulaci průchodu jednoho paprsku objemem. Uvedená část kódu je paralelně prováděna pro více průchodů současně. V první části uvedeného kódu je nutné určit souřadnice aktuálního bodu obrazovky, ze kterého bude prováděno vrhání paprsku. Získaná souřadnice slouží, jako počátek, od kterého budou dále vrhány další paprsky v pořadí. Poslední úlohou získané souřadnice obrazovky je její využití při dopočítání indexů (*i0, i1, i2*), které slouží k přístupu do polí, které obsahují zobrazenou krychli bez předních stěn (*frontTexBuffer*) a bez zadních stěn (*backTexBuffer*). Získáváme tak vstupní bod parprsku (*frontHit*) a výstupní bod paprsku (*backHit*). Následně body normalizujeme a poté pomocí jednoduchého vzorce získáváme směrový vektor (*dir*). Po získání směrového vektoru určíme jeho velikost (*sqrLen*) a v případě, že je dostatečně velká, provedeme výpočet délky trasy, kte-

rou bude paprsek procházet po jednotlivých krocích. Velikost vektoru nám říká, zda lze zanedbat průchod paprsku nebo je nutné simulaci provádět. Pokud byla velikost dostatečné velká a známe již délku trasy paprsku, provedeme normalizaci směrového vektoru a vrháme paprsek do objemu *(metoda castRay)*. Úlohou metody *castRay* je provádění kompozice pro každý krok, který urazí paprsek po trase, jejíž délka a směr byly předány, jako argumenty funkce. Velikostí kroku určujeme úroveň vzorkování. Čím je hodnota *stepSize* menší, tím vyšší je úroveň vzorkování.Zde je nutné upozornit na negativní dopad, který může být způsoben příliš nízkou hodnotou kroku vzorkování, která zpomaluje celkový proces vizualizace. Je tedy nutné určit optimální hodnotu, která bude podávat vizuálně dobré výsledky a současně přijatelnou dobu trvání vykreslování scény.

Ukázka výstupu vymodelovaných objektů pomocí algoritmu RayCasting:

Obrázek 22: 3D model humra vizualizovaný pomocí algoritmu RayCasting

Obrázek 23: 3D model humra vizualizovaný pomocí algoritmu RayCasting v barevném podání

5.6 Obecné vlastnosti modulů

Pro oba implementované moduly byla vytvořena uživatelská příručka a také programátorská dokumentace. Součástí programátorské dokumentace jsou rovněž třídní diagramy zachycující moduly, jako celek. Moduly jsou vytvořeny podle návrhového vzoru MVC (Model View Controller), který umožňuje oddělit logiku aplikace od uživatelského rozhraní. Způsob ovládání uživatelského rozhraní obou modulů je popsán v samostatné uživatelské příručce, která je součásti práce. Moduly jsou na sobě navzájem nezávislé a lze je provozovat odděleně. Pro externí knihovnu JOGL byl vytvořen samostatný modul, který obsahuje pouze knihovnu samotnou. Výhodou vytvoření samostatného modulu je možnost budoucího využití knihovny dalšími moduly. Současně je modul knihovny JOGL vyžadován, jako povinná závislost modulu vizualizace objemové grafiky. Moduly spolupracují s interní logikou systému FOTOM^{NG}, který umožňuje načítat soubory z disku, tyto soubory upravit pomocí existujících nástrojů pro práci s obrazem a následně předat zpracovaná data dále ke zpracování ostatním modulům. Tato možnost načítání dat je zohledněna a je tak možné do modulů takto zpracované snímky načítat a dále s nimi pracovat. Součástí uživatelského rozhraní obou modulů jsou okna zobrazující aktuální informace o načtených datech (v terminologii platformy NetBeans se jedná o manažerská okna). V případě modulu Radonovy transformace jsou zobrazeny informace o jednotlivých řezech v aktuálně zobrazeném snímku, úrovni nastavené prahové hodnoty pro daný snímek a obecné vlastnosti týkající se snímku. V modulu vizualizace objemových dat jsou naopak zobrazeny informace o aktuálně načtených objemových datech. Mezi tyto informace patří údaj o velikosti objemu, plocha průřezu a další informace popisující objemová data.

6 Zhodnocení dosažených výsledků

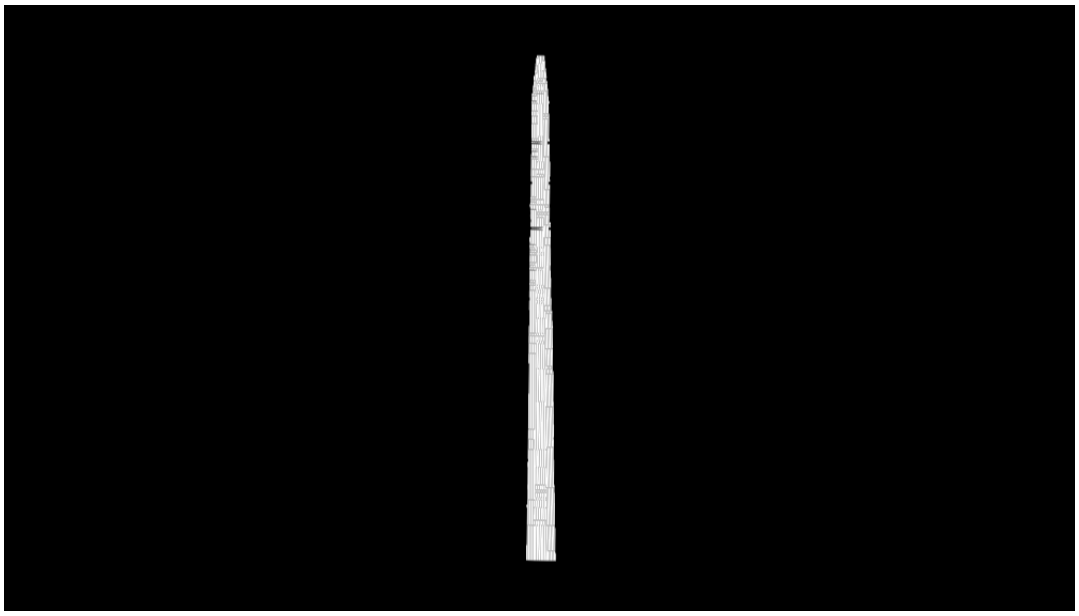
Na základě teoretických východisek, definovaných na počátku práce, byly úspěšně sestrojeny dva nové moduly pro systém FOTOM^{NG}. První modul je určen pro využití algoritmu inverzní Radonovy transformace k rekonstrukci objektů zájmu a druhý modul slouží pro vizualizaci objemových dat pomocí vybraných vizualizačních technik. Během práce byly implementované moduly prakticky vyzkoušeny na společném projektu s Fakultou stavební VŠB-TU Ostrava. Konkrétní výsledky spolupráce jsou popsány v následující podkapitole.

6.1 Výsledky společného projektu s Fakultou stavební VŠB-TU Ostrava

Záměrem projektu bylo porovnání konvenčních geodetických postupů při měření deformací objektu zájmu, spolu se softwarově provedeným měřením, které je prováděno pouze nad sérií snímků nafoceného objektu zájmu. Vybraným objektem zájmu byl zvolen sloup vysokého napětí, jehož ocelová konstrukce podléhá deformaci vlivem pohybu podloží, pracujících spojů a vlivem tahu vodičů, které jsou na sloup připojeny. V první fázi projektu byly stanoveny kontrolní body na sloupu, které posloužily jako referenční body pro geodetické měření. Kontrolních bodů bylo učeno celkem pět. Ve stejných bodech byly provedeny řezy na poskytnutých snímcích. Jelikož sloup představuje vysoký objekt, pořízené fotografie podléhají perspektivnímu zkreslení. Vrchol sloupu, který je vysoko a je od pozorovatele dále, se nám jeví menší, než jeho podstava, která je k pozorovateli blíže. V ideálním případě by mělo být snímací zařízení při pořizování snímků kolmé vůči snímanému objektu. Toho lze docílit velkou vzdáleností snímacího zařízení od snímaného objektu, avšak za cenu snížení detailů. Toto řešení je v případě měření nepřípustné. Pro řešení uvedeného problému byl spoluřešitelem sestrojen nový modul, který umožňuje provést náklon snímku v ose z. Pomocí tohoto modulu byly následně upraveny všechny snímky, kde docházelo k výraznému perspektivními zkreslení. Úkolem implementovaného modulu inverzní Radonovy transformace bylo provedení zpětné rekonstrukce objektu zájmu v daných řezech. Vytvořené filtrované snímky byly dále použity k rekonstrukci sloupu pomocí existujícího modulu pro 3D modelování v systému FOTOM^{NG}. Pro 3D modelování bylo nutné určit polygonální tvar objektů na všech filtrovaných snímcích. Tento úkol byl proveden pomocí dostupných nástrojů systému FOTOM^{NG}, které umožňují definovat body tvořící polygon, který geometricky popisuje vytvořený objekt. Na základě pěti polygonů byl pomocí modulu 3D modelování sestrojen 3D model sloupu. Poslední fází projektu bylo softwarové přeměření náklonu sloupu vlivem deformací a porovnání výsledné hodnoty spolu s výsledkem geodetického měření. Měření bylo provedeno pomocí modulu 3D modelování, který kromě vytvořeného 3D modelu poskytuje i údaje o zrekonstruovaném objektu.

I přesto, že poskytnuté snímky sloupu vysokého napětí nesplňovaly ideální podmínky postupu pořízení, výsledek porovnání naměřených hodnot spolu s numerický získanými hodnotami dopadl uspokojivě. Skutečné hodnoty vychýlení konstrukce od projektované hodnoty, naměřené pomocí geodetických postupů, se pohybovaly v rozmezí 162 mm až 346 mm. Ve stejném rozmezí se pohybovali i naměřené hodnoty pomocí

modulů systému FOTOM^{NG}. Z výsledků projektu vyplývá, že byl realizován velmi efektivní způsob, kterým lze pomocí numerických metod prováděných pouze nad dostupnými snímky objektu zájmu provést analýzu deformačního stavu objektu. Na základě těchto výsledků byl napsán článek, který bude prezentován na mezinárodní konferenci SGEM 2015, která se koná v období od 16.6.2015 do 25.6.2015 v Bulharsku. Dále bude tento článek prezentován na Mezinárodní konferenci Geodézie a Důlního Měřictví 2015, která se koná v období od 24.6.2015 do 26.6.2015 v Praze. Článek popisuje detaily měření spolu s popisem realizace a výsledky samotného projektu. Na závěr projektu, jako doplňující vizualizační část, byly na sloupu provedeny dodatečné řezy (celkem 200), které byly využity k rekonstrukci sloupu pomocí druhého modulu, který zobrazil sloup z objemových dat. Výsledný snímek vizualizace můžeme vidět na obrázku číslo 24.



Obrázek 24: Rekonstrukce sloupu vysokého napětí pomocí druhého modulu vizualizace objemových dat z 200 řezů

V horní třetině snímku můžeme vidět dvě místa, kde je kontura 3D modelu sloupu přerušena. K tomuto jevu dochází v situaci, kdy v daném řezu není správně zrekonstruován objekt pomocí inverzní Radonovy transformace. Vybrané řezy nebyly správně detekovány na hranicích objektu ve všech snímcích z dostupné série. Řešením této situace je opravení chybných řezů na snímcích a opětovné provedení inverzní Radonovy transformace. I přes to, že objemová data (série snímků zrekonstruovaného objektu v řezech) obsahují deformované řezy, lze úspěšně zrekonstruovat 3D model objektu. V případě modelování pomocí objemové grafiky není nutné, aby byly všechny zrekonstruované řezy objektu v ideální podobě. Rekonstrukci 3D objektu lze provést i s poškozenými řezy. V případech, kdy jsou zrekonstruované řezy potřeba pro definování polygonálního tvaru objektu, je nutné tyto řezy opravit a provést opětovnou zpětnou rekonstrukci. Výsledný

modul pro modelování objemové (voxelové) grafiky je v praxi použitelný a lze jej použít i na rekonstrukci objektů, které nespadají pouze do oblasti zdravotnictví.

7 Závěr

V rámci realizace diplomové práce byly splněny všechny body zadání. Výstup práce lze rozdělit do dvou částí. První část je tvořena teoretickým popisem postupů a východisek vedoucích k řešení dané problematiky. Druhá část je tvořena praktickou realizací dvou nových modulů systému FOTOM^{NG}, jejichž návrh je založen základě teoretického výstupu první části.

První z vytvořených modulů implementuje algoritmus inverzní Radonovy transformace spolu s vybranými algoritmy zpracování obrazu. Záměrem modulu je použití inverzní Radonovy transformace k rekonstrukci vybraného objektu zájmu v řezech, které jsou automaticky vytvářeny v sérii snímků objektu. Funkcionalita modulu byla úspěšně otestována použitím modulu na praktickém projektu, který se uskutečnil ve spolupráci s Fakultou stavební VŠB-TU Ostrava, z jehož výstupu byl napsán článek pro prezentaci na mezinárodní konferenci SGEM 2015 a Mezinárodní konferenci Geodézie a Důlního Měřictví 2015. Popis a výstup projektu byl popsán v teoretické části práce. Modul využívá při automatické tvorbě řezů algoritmy segmentace obrazu, doplněné o algoritmy matematické morfologie, k detekci objektu zájmu na snímcích. Výstup dat prvního modulu může být použit, jak vstup pro druhý modul, který data vizualizuje v 3D podobě.

Druhý realizovaný modul je určen pro vizualizaci objemové (voxelové) grafiky z objemových dat, která jsou představována sérií snímků řezů objektu. Modul implementuje dva vizualizační algoritmy, pro jejichž implementaci byla využita grafická knihovna Java OpenGL (JOGL). Prvním implementovaným algoritmem je algoritmus 3D Texture Slicing, druhým algoritmem je algoritmus RayCasting. Dále modul umožňuje filtraci objemových dat spolu s možným vykreslením v barevné podobě. Modul byl otestován s úspěšnými výsledky vizualizace na dostupných snímcích počítačové tomografie a taktéž na snímcích, které byly vytvořeny v rámci spolupráce na uvedeném projektu.

Nově implementované moduly byly začleněny do systému FOTOM^{NG} a rozšířili tak jeho funkcionalitu. V průběhu realizace modulů byl kladen důraz na optimalizaci výkonu použitím paralelního zpracování vykonávaných úloh. Vybrané části algoritmů jsou navrženy a implementovány pro paralelní zpracování na více jádrových procesorech, s cílem využití maximálního dostupného výkonu. Pro oba moduly byla vypracována uživatelská dokumentace s popisem ovládání a také programátorská dokumentace s popisem implementovaného kódu.

Jako doporučená možnost budoucího rozšíření je implementace dalších modulů pro práci s objemovými daty, zaměřených na jejich úpravu, segmentaci a případnou detekci vybraných objektů zájmu. Dále by bylo vhodné zvážit budoucí možnost využití výkonu moderních grafických k paralelnímu zpracování kritických částí kódu vizualizace objemových dat.

8 Reference

[1] Vlček, Vítězslav Vít a Segeth, Karel: *Matematika dokonale ukrytá v počítačové tomografii. Pokroky matematiky, fyziky a astronomie*, Vol. 53 (2008), No. 3, 199—210, Citováno: 24.11.2014, Dostupné na: http://dml.cz/dmlcz/141859

[2] Ptáček, Jaroslav, *Filtrovaná zpětná projekce - rozšířený popis*, Univerzita Palackého v Olomouci, 2013, Citováno: 27.11.2014, Dostupné na: http://goo.gl/EBCzdF

[3] Kubíček, Jan, *Radonova transformace*, VŠB-TU - Ostrava, Fakulta biomedicínského inženýrství, Citováno: 24.11.2014, Dostupné na: http://www.cs.vsb.cz/licev/lzs%20I_cviceni/ct_rekonstrukce.pdf

[4] Sojka, Eduard, *Digitální zpracování a analýza obrazů*, VŠB-TU - Ostrava, Fakulta elektrotechniky a informatiky, 2000, ISBN: 80-7078-746-5, Citováno: 10.1.2015, Dostupné na: http://mrl.cs.vsb.cz/people/sojka/dzo/digitalni_zpracovani_obrazu.pdf

[5] Callahan, Steven P., Callahan, Jason H., Sheidegger, Carlos E., Silva, Cláudio T. *Direct Volume Rendering: A 3D Plotting Technique for Scientific Data*, říjen 2007, Citováno: 25.1.2015, Dostupné na http://www.sci.utah.edu/~cscheid/pubs/dvr.pdf

[6] Geršl, Vladimír, *Modelování a vykreslování objemových mraků pro herní aplikace*, Západočeská univerzita v Plzni, 2010, Plzeň, Citováno: 30.1.2015, Dostupné na: http://graphics.zcu.cz/files/100_DP_2010_Gersl_Vladimir.pdf

[7] Ikits, Milan, Kniss, Joe, Lefohn, Aaron, Hansen, Charles, *Chapter 39. Volume Rendering Techniques*, NVIDIA Developer Zone, 2004, NVIDIA Corporation, Citováno: 2.2.2015, Dostupné na: http://http.developer.nvidia.com/GPUGems/gpugems_ch39.html

[8] Krátký, Michal, *Databázové a informační systémy 2 - Prostorová data a databázové systémy*, prosinec 2014, FEI - VŠB-TU Ostrava, Citováno: 5.2.2015, Dostupné na: http://dbedu.cs.vsb.cz/SubPages/OpenFile.aspx?file=2014-2015/dais2/student/dais2-13b.pdf

[9] Platoš, Jan, *Komprese dat 4 - Komprese obrazu, videa a zvuku*, duben 2008, VŠB - TU Ostrava, Citováno: 7.2.2015, Dostupné na: http://homel.vsb.cz/~pla06/files/kod/KOD_4.pdf

[10] Pelikán, Josef, *Visualizace objemových dat*, 1996-2009, CGG MFF UK Praha, Citováno: 8.2.2015, Dostupné na: http://cgg.mff.cuni.cz/~pepca/lectures/pdf/volume.pdf

[11] Crawfis, R., *Texture-based Volume Rendering*, prosinec 2003, Ohio State University, Citováno: 9.2.2015, Dostupné na: http://web.cse.ohio-state.edu/~crawfis/cis694L/Slides/TextureSlicing.pdf

[12] Wikipedia, *Java OpenGL*, poslední aktualizace 23. prosince 2014, Citováno: 14.2.2015, Citováno: 9.2.2015, Dostupné na: http://en.wikipedia.org/wiki/Java_OpenGL

[13] Heiko Böck, *NetBeans Podrobný Průvodce Programátora*, 2010, Computer Press, Czech Republic, ISBN: 978-80-251-3116-9, 320 Pages

[14] Java Platform SE 7, *Class ForkJoinPool*, 2014, Oracle, Citováno: 15.2.2015, Dostupné na: http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ForkJoinPool.html

A Přílohy na přiloženém CD

A.1 Příloha 1

Text diplomové práce v elektronické podobě

A.2 Příloha 2

Zdrojové kódy modulů + modul s knihovnou JOGL

A.3 Příloha 3

Uživatelská dokumentace

A.4 Příloha 4

Programátorská dokumentace